TSX-Plus
System Manager's
Guide

s&h computer systems, inc.

TSX-Plus
System Manager's Guide

Fourth Edition
First Printing -- September, 1984

S&H Computer Systems, Inc.
1027 17th Avenue South
Nashville, Tennessee USA
37212-2299
(615)-327-3670

CONTENTS

Chapter 4

Chapter 5

Chapter 6

Chapter 7

Appendix A

Appendix B

INTRODUCTION

The purpose of the TSX-Plus System Manager's Guide is to provide information necessary to manage the system resources for the TSX-Plus operating system. It is intended to provide more detailed information on the internal operation of TSX-Plus for people who are already familiar with the features provided. See the TSX-Plus Reference Manual for information on the features provided by TSX-Plus.

Chapter 1 - System and File Access Security
The system manager can impose certain restrictions on system use and file access by selecting available security options. Users or lines can be: locked to a program; limited in access to devices or files; or restricted to a maximum priority for program execution. Individual users may be given command privilege.

Chapter 2 - Account Authorization Program
An account authorization program may be used by the system manager to grant access to the system by initializing valid accounts with passwords. The facility allows the system manager to determine for each account use of virtual lines and detached jobs, operator privilege and the maximum job execution priority. Execution of a start-up command file can also be specified which may contain system and file access security restrictions.

Chapter 3 - System Overview
An understanding of the internal operation and organization of TSX-Plus provides the system manager with the the basic knowledge necessary for optimizing system performance. The system overview provides information concerning memory organization (with a detailed map of the TSX-Plus physical memory layout), I/O mapping, and execution scheduling (including a job scheduling flow diagram and algorithm).

Chapter 4 - Device Handlers
The information in this chapter provides the system manager (and system programmers who wish to write special device handlers) an understanding of the extensions and restrictions imposed on device handlers in the TSX-Plus environment. Device handler debugging and use of special device handlers (CL, IB, and VM) are discussed.

Chapter 5 - Terminal I/O
The information in this chapter provides the system manager with an understanding of the internal operation of the terminal and communication line (CL) handler. Modem control and the RS232 pin connection required for phone support is outlined.

Chapter 6 - System Tuning
With the basic knowledge of the organization and operation of TSX-Plus, the system manager can utilize various tools to better optimize the TSX-Plus execution environment. Suggestions concerning optimization for memory, I/O, and execution scheduling are provided.

Chapter 7 - SYSMON - Dynamic System Display Utility

The SYSMON utility displays information about system activities and resources. This utility can help the system manager gain more information about the specific environment to facilitate resource optimization.

Appendices

Appendix A describes the error messages which can be generated by TSX-Plus when it is being started (R TSX). Appendix B describes the fatal system error messages generated when abnormal conditions occur during operation of TSX-Plus.

1. SYSTEM AND FILE ACCESS SECURITY

TSX-Plus provides a number of system security options that allow the site manager to control access to the system by time-sharing users. By selecting the appropriate combination of options the system manager can control who can log onto the system, which files or devices each user can access and can also lock users to application programs. There are six facilities that can be used to control system access:

1. Start-up command files.

2. Log-off command files.

3. The RUN/LOCK switch.

4. The ACCESS command.

5. The SET MAXPRIORITY command.

6. Operator privilege.

7. The LOGON and account authorization programs.

In addition to these security features, TSX-Plus also provides a use accounting facility that keeps track of the number of time-sharing sessions and the total connect time for each user.

1.1 Start-up command files

When TSX-Plus is generated, the system manager may specify for each time-sharing line the name of a command file that is to be executed each time the line is started. The command file name is specified by using a "CMDFIL" macro within the line definition block in TSGEN. Different command files may be specified for each line and any or all lines may be generated without start-up command files.

If a line has a start-up command file, that command file is started each time the line is initialized (e.g., when a user presses carriage return on an inactive line). Start-up command files are different than other command files in that their execution cannot be aborted by typing control-C. This allows the system manager to place any desired commands in a start-up command file to be executed to completion regardless of the actions of the time-sharing user. However, if the command file aborts for some other reason, the line may be granted access to the system without proper initialization. This may be avoided by disabling command file aborts, except in the most serious circumstances, by setting the error abort level as the first command. This is expecially important for lines started with complex command files and for dial-up lines. For example:

```
SET ERROR FATAL
      .
      .
      .
R/LOCK LOGON
OFF
```

This would prevent the line from accessing the system even if the LOGON program were not found.

A start-up command file can contain any keyboard command and can run one or more programs. Control-C resumes its normal function when the start-up command file is terminated or a program initiated by it requests input from the terminal. It is suggested that start-up command files be given the extension "TSX" to prevent their being tampered with by users who do not have operator privilege (see below). Note, if "TSX" is used as the file extension, it must be specified with the file name in the CMDFIL macro as the default extension is "COM". The default device is "SY:".

The listing of a start-up command file can be suppressed by placing the two character sequence "^(" at the front of the command file.

1.2 Log-off command files

It is possible to declare a command file that is to be executed when a job logs off. To declare a log-off command file, place a command of the following form in the start-up command file for the job:

```
SET LOGOFF FILE=name
```

Where "name" is the file specification for the log-off command file. The SET LOGOFF command is only legal within the start-up command file for the job. The log-off command file is executed whenever the job logs off. Be careful with what you put in a log-off command file since the execution of a log-off command file cannot be aborted by typing control-C. The listing of a log-off command file can be suppressed by placing "^(" as the first two characters of the file.

1.3 The RUN/LOCK switch

The "R" and "RUN" commands accept a "/LOCK" switch that causes the program being run to be "locked" to the time-sharing line. A locked program executes in the normal fashion, and may chain to other programs (which become locked). However, if a locked program exits or is aborted by typing control-C the line is automatically logged off. Note that one can prevent an ongoing program from being aborted by control-C by doing an .SCCA EMT or by using the TSX-Plus "D" program controlled terminal option (see the TSX-Plus Reference Manual for information on defining activation characters).

In a situation in which a time-sharing line is to be automatically locked to a program when the line is started, simply build a start-up command file for the

line and include as the last start-up entry in the file a "RUN/LOCK program" command.

1.4 The ACCESS command

The ACCESS keyboard command is used to limit access to devices and files. The ACCESS command is valid only if executed as part of a start-up command file.

The form of the ACCESS command is:

```
ACCESS dev:file.ext/switch,dev:file.ext/switch,...
```

Up to twenty "dev:file.ext" expressions may be specified. Each logical subset disk mounted also counts toward the limit of twenty entries in the access table.

If no ACCESS command is executed, the time-sharing user is allowed to access all devices and files on the system (with the exception of SYS and TSX-Plus files--see Operator Privilege, below). If any ACCESS command is executed, the user is restricted to accessing only the devices and files that are specified with the command.

The "dev:file.ext" expression has three items: the device name, the file name and the extension. The "*" (wildcard) character may be substituted for any or all of these three items. In this case the wildcard will allow access to any name that occurs in the wildcarded position. For example, "RK1:*.ABC" will allow access to any file on RK1 that has the extension "ABC". Consider the following ACCESS command:

```
ACCESS RK0:*.ABC,RK0:*.BAK,RK1:*.*,LP:
```

This allows access to any files on RK0 that have the extension "ABC" or "BAK"; it also allows access to all files on RK1 and LP. Note that the LP specification is needed if the user is to be allowed to access the line printer. Access privilege is needed to read, create, delete, or rename a file. A device can only be initialized (directory zeroed) if full access to the device is granted.

The ACCESS facility works by matching the user-specified device, file and extension names with those that were specified on the ACCESS command. This matching is done after any ASSIGNS of logical to physical device names are carried out.

Because the utility programs PIP, DUP and DIR directly access device directories, they exhibit minor deviations from expected access protection behavior. If access is granted to any files on a device, then DIR will be able to obtain the device directory. In order for PIP and DUP to access an individual file, the job must have at least /READ access to the full device, even if access has been granted to the specific file of interest. These deviations affect the DIR, COPY, TYPE, and PRINT commands among others.

The "/READ" switch may be specified with a device-file name to restrict access to the device-file to be read-only. For example, the following command allows full access to RK1 but read-only access to RK0.

```
ACCESS RK1:,RK0:/READ
```

Remember that the common utility programs, such as PIP and DIR, are required by most users and consequently at least SY:*.SAV/READ access is usually desirable. Also, access to the system library file (SY:SYSLIB.OBJ or SY:FORLIB.OBJ) and the system MACRO library file (SY:SYSMAC.SML) may be necessary for program development. Because of the limited number of ACCESS entries that may be made (20 for each job), it is not advisable to enumerate each specific file to which access is desired, but rather to cluster groups of files on the system disk or on logical subset disks. For example, the following ACCESS command could be used to grant full access to DL1 and limited access to the system disk:

```
ACCESS DL1:,SY:*.SAV/READ,SY:SYSLIB.OBJ/READ,SY:SYSMAC.SML/READ
```

The ACCESS and MOUNT commands can be used together to control access to logical subset disks. To control which logical disks are available to a user, specify the names of the files that contain the logical disks with the ACCESS command in the startup command file and then use MOUNT commands <u>after</u> the ACCESS command to associate logical disk units with the files. This will allow the user to access all files within the logical disk but will restrict access to other logical disks or files. For example, consider the following commands which could be placed in a startup command file:

```
ACCESS SY:/READ,DL0:CLASS1.DSK,DL0:CLASS2.DSK/READ
MOUNT LD1 DL0:CLASS1
MOUNT LD2 DL0:CLASS2
```

After executing this startup command file, the user will have read only access to all files on the system disk ("SY:"), read-write access to LD1 which is associated with the file DL0:CLASS1.DSK, and read-only access to LD2 which is associated with DL0:CLASS2.DSK. This will permit the user to initialize LD1 and create, edit, and delete files on LD1. The user may also create nested logical disks within LD1. Files on LD2 may be accessed for reading only.

1.5 The SET MAXPRIORITY command

TSX-Plus users can assign execution priority values to their jobs by use of the SET PRIORITY command and a TSX-Plus EMT. The maximum priority that a user is allowed can be controlled by use of either the TSAUTH program (in conjunction with the LOGON program), or the SET MAXPRIORITY command. Normally the TSAUTH program would be used to assigned maximum priorities if the LOGON facility is being used. The SET MAXPRIORITY command is intended primarily in situations where the LOGON facility is not being used but it is still desirable to limit the maximum authorized priority. In these cases the SET MAXPRIORITY command can be placed in the start-up command file for the line.

The form of the SET MAXPRIORITY command is:

SET MAXPRIORITY value

where "value" is in the range 0 to 127. The SET MAXPRIORITY command may only lower the maximum authorized priority value for the job, it may not increase it. Thus the system manager may restrict job priority by placing a SET MAXPRIORITY command in the start-up command file for a line.

1.6 Operator privilege

Certain system facilities and keyboard commands are only available to time-sharing users who are granted "Operator Privilege". The following list summarizes those system facilities that are restricted to users having operator privilege.

1. The $STOP, $SHUTDOWN and BOOT keyboard commands. If a user without operator privilege attempts to use one of these commands, TSX-Plus displays the message "?KMON-F-You're not privileged for that command." The $STOP, $SHUTDOWN and BOOT commands either halt the system or boot RT-11, depending on your system configuration. See the TSX-Plus Reference Manual for further information.

2. Real-time programming facilities such as the ability to access the I/O page and connect to real-time interrupts. Note that operator privilege is necessary to run the SYSMON utility, because it uses some real-time facilities.

3. Creation or execution of files with the extension ".TSX" or ".SYS".

4. Use of the TSAUTH account authorization program.

5. The ability to use the .PEEK and .POKE EMT's to access the I/O page or low memory areas.

6. The use of the EMT to set the user name (operator privilege is not required to determine the user name).

7. The ability to set the system date or time (DATE and TIME commands and the .SDTTM EMT).

8. The ability to issue SET TERMINAL commands for other lines.

9. Use of the SYSMON system status display program.

10. Use of the SETUP program on the Professional.

11. Use of the /IOPAGE switch with the RUN command.

Operator privilege can be granted in three ways. If the LOGON program is used, individual accounts can be granted or denied this privilege (see account authorization information in Chapter 2). If the LOGON program is not used, this privilege is specified on a line-by-line basis during TSX-Plus system generation by including $PRIV in the FLAGS macro. Note that even if the FLAGS macro for a given line includes the $PRIV flag and the LOGON program is run from that line, then operator privilege is still controlled by the account authorization system. A privilege user may also grant privilege to another line with the SET TERMINAL command.

1.7 Use of the LOGON facility

The TSX-Plus LOGON facility provides access security to the system by requiring users to enter a valid project-programmer number or user name and password before granting access to the system. In addition, the LOGON facility allows the system to grant different privileges to each user and provides system use accounting on a per user basis.

To use the LOGON facility the system manager must first use the account authorization program (see Chapter 2) to create an account authorization file. This file specifies the valid project-programmer numbers, user names, passwords, user start-up command file, and privileges. He must then generate a TSX-Plus system and specify a line-by-line start-up command file to be executed for each line that is to be forced to logon. The suggested name for this start-up command file is "SY:LOGON.TSX". This command file may contain any desired keyboard commands but should start by disabling error aborts, should lock the job to the LOGON program, and should end by logging the job off. In this fashion, the job will not be able to gain access to the system even if the LOGON program is missing or some other command fails. For example:

```
SET ERROR FATAL
      .
      .
      .
R/LOCK LOGON
OFF
```

This command causes the LOGON program to be started and "locked" to the line so that the user cannot run any other program until the logon has been successfully completed. Note that the logon program (LOGON.SAV) must be present on the system device. The OFF command will only be executed if the LOGON program cannot be run.

Note that for each job there may be two start-up command files: the first is specified with the CMDFIL macro in TSGEN and is associated with a physical time-sharing line; the second is associated with a particular user (account name, project-programmer number) and is invoked through the LOGON program and account authorization system.

To prevent listing the start-up command file, the character sequence "^(" can be placed at the beginning of the command file. Thus, the logon start-up file for a physical time-sharing line might contain:

```
^(SET ERROR FATAL
R/LOCK LOGON
OFF
```

A SET LOGOFF command can be placed in the start-up command file to declare the name of a command file to be executed when the job logs off.

2. ACCOUNT AUTHORIZATION PROGRAM

TSAUTH, the TSX-Plus account authorization program, is used to authorize project-programmer numbers for access to the system when the LOGON facility is used. It is also used to display the use accounting statistics that are collected by the LOGON facility.

A user must have operator privilege to run TSAUTH under TSX-Plus. However, TSAUTH may also be run directly under RT-11 without TSX-Plus. In a hostile environment it might be desirable to restrict access to the TSX-Plus distribution media and to keep the TSAUTH program on a removable medium rather than keeping it on the system disk. TSAUTH creates a file on SY named "ACCESS.TSX". Note that operator privilege is required to create or execute any file with the extension "TSX".

Whenever TSAUTH is started it checks to see if an account authorization file already exists. If not it prints the message:

```
Cannot open account authorization file "SY:ACCESS.TSX"
Do you want to initialize a new authorization file?
```

If you respond "YES" (or "Y") to this question it will ask you how many project-programmer numbers (PPN's) you want to reserve room for in the file. Respond by entering the maximum number of accounts that you anticipate ever needing to have authorized at any one time. As old accounts are deauthorized, file space is recovered that can be used for new accounts. Note however that the only way to enlarge the ACCESS file is to delete it and build a new, larger one from scratch. Do not underestimate the potential number of accounts desired.

The format of the account authorization file changed with TSX-Plus version 4.0. If your authorization file was created with an earlier version of TSAUTH, the AUTCVT program, which is described at the end of this chapter, must be used to convert the file to the new format.

Once the authorization file is found or built, TSAUTH prints an asterisk indicating it is waiting for a command. Commands are entered as a single letter followed by a space and (for most commands) a user name or a project-programmer number typed with a comma separating the project number from the programmer number. The L(IST), K(ILL), U(SE) and R(ESET) commands allow a wildcard asterisk character to be used in place of the project number, programmer number or both. This allows sets of project-programmer numbers to be dealt with as a group. These commands also allow use of the user name instead of the PPN, allowing one to do these operations without knowing the PPN.

TSX-Plus gives no special significance to the grouping of accounts by project and programmer numbers. However, for convenience in using the TSAUTH program, it is suggested that a common project number be used for all PPN's associated with the same project or class and the programmer number be used to identify an individual.

User names given to accounts should be unique. As TSAUTH and LOGON read the ACCESS.TSX file sequentially, only the first occurrence of a given user name will be recognized at logon time or during TSAUTH file updating. For example, if two users have the user name SMITH and the one with the later record in the access file attempts to log on by using his name, LOGON will not recognize his password, as it is expecting the password for the first SMITH. However, he can still log in by specifying his project-programmer number, since PPN's must be unique.

2.1 Command summary

The following table lists a brief summary of the commands accepted by TSAUTH:

Command	Function
A proj,prog	Authorize a new account
K user-name	De-authorize an existing account
L user-name	List authorization status
U user-name	List account usage statistics
C	Create charge file (DK:CHARGE.TSX)
R user-name	Reset account usage statistics
E	Exit TSAUTH

Commands which accept a user-name will also accept project-programmer numbers or wildcards. The A command accepts only PPN's.

2.2 Authorizing a project-programmer number

The "A" (Authorize) command is used to add a new PPN to the authorization file. The form of this command is:

A proj,prog

A wildcard ("*") may not be substituted for either the project or programmer number. A project-programmer number must be deauthorized by the use of the kill command before it can be reauthorized. Project and programmer numbers must be decimal values in the range 1 to 65535.

TSAUTH responds to the "A" command by asking a series of questions as follows:

a) User Name:

Enter the user name to be associated with the PPN as a 1 to 12 character alphanumeric string. User names may be composed of letters and digits but may not contain spaces. The user name must be unique to the PPN; you may not have more than one PPN with a given user name. This is necessary as you can log in with the user name as well as with the PPN.

b) Password:

Enter a 1 to 8 character alphanumeric string that is the password to be associated with the PPN.

c) Start-up file:

Enter the name (dev:file.ext) of the start-up command file to be executed whenever this user logs on. Press return if no startup command file is desired. The default device is SY and the default extension is COM.

Note that for each job, two startup command files may be executed; the file associated with the line and the file associated with the user. Usually, the start-up command file associated with the line will contain little more than R/LOCK LOGON, whereas logical assignments and ACCESS commands will be located in the command file associated with the user, as defined by this parameter.

d) Virtual lines:

Respond with "Y" for yes or "N" for no, indicating whether the account is to be allowed to use the TSX-Plus virtual line facility.

e) Detached jobs:

Respond Yes/No indicating whether the account is to be allowed to use detached jobs.

f) Operator commands:

Respond Yes/No indicating whether the account has operator privilege. Generally, this should be restricted to a small set of users.

g) Maximum execution priority:

Respond with a (decimal) number in the range of 1 to 127 to restrict this user's maximum job execution priority. The default value is 50.

Example: In the following example a PPN 107,423 is authorized with the user name "DAGWOOD" and the password "SECRET" and the start-up command file named "SY:SU107.TSX" is specified. The maximum execution priority defaults to 50 since no value is entered.

```
*A 107,423
User name:DAGWOOD
Password:SECRET
Start-up file:SY:SU107.TSX
Virtual lines:Y
Detached jobs:N
Operator commands:N
Maximum execution priority:
*
```

2.3 Deauthorizing accounts

The "K" (Kill) command is used to deauthorize project-programmer numbers. The form of this command is:

K proj,prog

or

K user-name

where a wildcard character ("*") may be substituted for the project number, programmer number or both.

Examples:

1. Deauthorize PPN 107,423.

```
*K 107,423
```

2. Deauthorize all PPN's with the project number 237.

```
*K 237,*
```

3. Deauthorize a PPN with the user name DAGWOOD.

```
*K DAGWOOD
```

2.4 Listing account status

The "L" (List) command is used to list the current authorization status of any or all accounts. The form of this command is:

L proj,prog

or

L user-name

The wildcard character may be substituted for the project and programmer numbers. The information listed includes all of the items that were specified when the account was authorized.

Example:

List the current status of the account with user name DAGWOOD.

```
*L DAGWOOD

PPN:107,423
user-name:DAGWOOD
Password:SECRET
Start-up file:SY:SU107.TSX
Virtual lines:Y
Detached jobs:N
Operator commands:N
Maximum execution priority:50
*
```

2.5 Listing account usage statistics

The "U" (Usage) command can be used to display the account usage statistics which consist of the number of sessions, the connect time, and the CPU time. The form of this command is:

```
U proj,prog
```

or

```
U user-name
```

The wildcard character may be substituted for the project number, programmer number, or both.

Example:

List the usage statistics for all accounts with the programmer number 423.

```
*U *,423

PPN:107,423     # sessions=14 Connect time=01:23:00  CPU=00:03:07.4
PPN:413,423     # sessions=5  Connect time=14:02:00  CPU=01:13:02.5
PPN:21,423      # sessions=10 Connect time=01:11:00  CPU=00:49:18.9
```

2.6 Creating a charge information file

The "C" (Charge) command causes TSAUTH to create a file of usage information. The file is named "DK:CHARGE.TSX"; it contains one record for each PPN; each record is terminated with a carriage return and line feed.

The format of a charge record is as follows:

Columns	Contents
1	(blank)
2 - 6	Project number
7	(blank)
8 - 12	Programmer number
13	(blank)
14 - 18	Number of logons
19	(blank)
20 - 24	Number of minutes of connect time
25	(blank)
26 - 33	CPU time used (0.1 second units)
34	(blank)
35 - 46	User-name (left justified and padded with blanks)
47	(carriage return)
48	(line feed)

2.7 Resetting account usage statistics

The "R" (Reset) command resets the account usage statistics (number of sessions, connect time, and CPU time) to zero for all or a selected set of accounts. The form of this command is:

R proj,prog

or

R user-name

where the wildcard character may be substituted for the project number, the programmer number, or both.

Examples:

1. Reset all PPN's with the project number 21.

   R 21,

2. Reset all PPN's.

   *R *,*

3. Reset the PPN with the user name DAGWOOD.

   *R DAGWOOD

2.8 Exiting from the account authorization program

The "E" (Exit) command is used to exit from the TSAUTH program to the keyboard monitor. The form of this command is:

```
E
```

2.9 AUTCVT program

The TSAUTH and LOGON programs supplied with TSX-Plus are incompatible with authorization files created prior to version 4.0. A conversion program named AUTCVT is supplied to convert old format authorization files to the new format. It is suggested that a copy of the old format file be made before the conversion is performed since there is no way to convert back to the old format.

The AUTCVT program reads an old format authorization file with the name SY:ACCESS.TSX and creates a new format authorization file with the same name (superseding the old file). This conversion should be done while running under RT-11 before starting TSX-Plus. In the process of converting the file, AUTCVT prints each project-programmer number and requests a corresponding user name to be entered. If you do not wish to specify a user name for some PPN's, press return without entering a name. This will limit the user to logging on using his PPN. If a user name is entered, then either the user name or the PPN may be used in subsequent logons.

3. SYSTEM OVERVIEW

This chapter presents an overview of the TSX-Plus system organization and operation. It is intended to provide background information for users who want to know more about the system internal organization and operation.

3.1 Memory organization

Memory is organized into two major divisions: memory used by the operating system and memory available for user programs. The memory required by the operating system is permanently allocated and contains both code regions and data structures reserved for its exclusive use. In contrast, the content of user memory changes frequently as different jobs are swapped in and out of memory. Associated with each job, the system maintains a 4K byte job context region. Job swapping only occurs when a user job needs service and there is not enough contiguous free memory to load it and its job context region. Job swapping may be disabled entirely as a system generation option. In this case, a new job can only be started when sufficient user memory is already available.

3.1.1 System memory mapping

The operating system is divided into four distinct regions: kernel root, system overlays, mapped data, and the I/O page. The kernel root is mapped using kernel PARs (page address registers) 0 through 4. Because of this, the kernel root code region is restricted to a maximum of 40K bytes. (Each PAR maps 8K bytes.) The I/O page is mapped through kernel PAR 7.

Each system overlay code region and mapped device handler is accessed through kernel PAR 5 and is therefore restricted in size to a maximum of 8K bytes. Only one memory resident overlay or handler code region may be mapped at a time.

Each mapped data region is an individual storage area mapped through kernel PAR 6. Because of this, each data region is restricted in size to a maximum of 8K bytes. Only one data region may be accessed at a time.

The following diagram illustrates the virtual address organization of TSX-Plus during execution.

Virtual Memory in the TSX-Plus Kernel

```
+---------------------------+  177777
|         I/O Page          |
+---------------------------+  160000
|  Mapped Data Regions      |
+---------------------------+  140000
|    System Overlay or      |
|  Mapped Handler Regions   |
+---------------------------+  120000
|                           |
|                           |
|    Kernel Root Code       |
|     and Data Region       |
|                           |
|                           |
+---------------------------+  0
```

3.1.1.1 Kernel root: The kernel root contains: device handler vectors (located from zero to octal 500); the memory resident overlay handler and tables necessary for interfacing to overlay code sections; data tables allocated in TSGEN; executive code including the job swapper, execution scheduler, etc.; I/O related processing code; clock and terminal interrupt entry code; generalized data cache code; and the startup initialization code. To conserve space, TSX-Plus re-uses the memory containing the startup initialization code by allocating data structures which do not require initialization over it after completion of startup. If additional space is necessary, the top of TSX-Plus is extended. These buffers consist of the job information tables (simulated RMON); I/O queue elements; and system message buffers. Unmapped device handlers are loaded above these data buffers. The size of the entire kernel root region described here (including unmapped device handlers) must not exceed 40K bytes.

3.1.1.2 System overlay and mapped handler regions: There are currently fourteen memory resident overlay code regions. They are separated logically by function. Since only one overlay code region may be mapped at a time this functional separation reduces the number of calls to the overlay handler. Nine of the overlay code regions are optional and will only be loaded if the feature is selected in TSGEN. The functions performed by the overlay code regions are:

1. Terminal input and output operations
2. Programmed EMT requests code
3. Programmed EMT requests additional code
4. Directory manipulation requests and directory cache buffers
5. Miscellaneous executive functions such as clock processing and fatal error processing
6. * Program logical address space requests (PLAS)
7. * Device spooling with buffers
8. * Record locking and data structures
9. * Message communication and data structures
10. * Real-time service requests
11. * Mapped I/O servicing
12. * Single line editor
13. * Communication line handler
14. * User program debugger

* Denotes optional overlays that are only loaded into memory if the corresponding feature is selected during system generation.

The number of mapped handlers will depend on the device declarations (DEVDEF) in TSGEN and the corresponding attributes declared or imposed by the initialization routine for each device handler.

3.1.1.3 Mapped data regions: The mapped data regions allocated during startup contain the memory map table; the terminal input and output character buffers, and the following optional buffers: shared data cache buffers, mapped I/O buffers, performance monitor buffers, and generalized data cache buffers.

3.1.1.4 Shared run-time systems: In addition to the system regions described, a reserved memory region, also pre-allocated by the system, contains user-defined shared run-time systems such as those provided with COBOL-Plus and DBL.

3.1.2 Physical layout of TSX-Plus

The kernel root begins at physical memory address zero. Its size is variable, depending on options selected during system generation, and may extend up to 40K bytes. All of the mapped data regions are allocated directly above the kernel root with the exception of the generalized data cache buffers which are allocated directly below the system overlay regions and any optional shared run-times. The system overlay regions are allocated at the top of physical memory, or at the top selected by the MEMSIZ parameter if not all physical memory is to be used by the system. For example, some portion of memory may be reserved for use by a memory based disk emulator such as VM. Shared run-time systems, if any, are loaded directly below the system overlay regions as are mapped device handlers. Data buffers used by the generalized data caching facility are allocated below any mapped device handlers. Finally, all the physical memory between the two memory areas allocated by the operating system is available for time-sharing users. The following diagram depicts the physical memory allocation of TSX-Plus during execution:

Physical Memory Use by TSX-Plus

```
+----------------------------+
|          I/O Page          |
+----------------------------+
~                            ~

~                            ~
+----------------------------+  Top of physical memory
|  VM Pseudo-disk Data Area  |
|        (optional)          |
+----------------------------+  Top of TSX-Plus (MEMSIZ)
|   System Overlay Regions   |
|      (some optional)       |
+----------------------------+
|     Shared Run-Times       |
|        (optional)          |
+----------------------------+
|   Mapped Device Handlers   |
+----------------------------+
|  Generalized Cache Buffers |
|        (optional)          |
+----------------------------+
|                            |
~                            ~
       User Job Region
~                            ~
|                            |
+----------------------------+
| Performance Monitor Buffer |
|        (optional)          |
+----------------------------+
|     Mapped I/O Buffers     |
|        (optional)          |
+----------------------------+
|    Terminal I/O Buffers    |
+----------------------------+
| Shared File Cache Buffers  |
|        (optional)          |
+----------------------------+
|      Memory Map Table      |
+----------------------------+  Maximum 40Kb
|  Unmapped Device Handlers  |
|  Initialization Code       |
|  Executive Code            |
|  TSGEN                     |
|  Overlay Tables            |
|  Interrupt Vectors         |
+----------------------------+  Physical 0
```

3.1.3 User memory

The user's job region, sandwiched between memory used for the operating system, is allocated dynamically, placing each user's job in the first available free memory area large enough to contain it. In a swapping system, each job can potentially be positioned anywhere within the region. A 4K byte job context region is appended immediately below each job image, allowing the job and its context region to be swapped together.

The virtual address space of each job is intrinsically limited to 64K bytes by the PDP-11 architecture, although the job may remap itself by use of real-time or shared run-time EMTs. In addition, each job may request and be granted more physical space by use of the PLAS requests. These extended memory regions may be used for virtual overlays or virtual arrays and need not be contiguous with the job's base image. When an extended job is swapped, the PLAS regions are swapped into a disk file separate from the base image.

3.2 I/O mapping

I/O mapping is a facility which allows DMA devices with 18-bit controllers or device handlers to be used with Q-bus systems with 22-bit address space.

The original LSI Q-bus used with 11/23 systems had 18 address lines allowing I/O transfers to take place within 256K bytes of memory. Device controllers developed during this period supported 18 address bits. With the introduction of the 11/23-Plus processor, four additional address bits were added to the Q-bus bringing the total to 22 address bits which allowed I/O transfers to take place to 4M bytes of memory. Unfortunately, many sites still have older device controllers that only support 18 bits and, in fact, DEC still does not build a Q-bus DY (RX02) controller that supports 22 bit DMA transfers. The 18-bit controllers will operate satisfactorily with 22-bit Q-bus systems provided that the I/O transfer is always within the lower 256K bytes of memory. This could cause problems with TSX-Plus since jobs may be located anywhere in physical memory and I/O transfers are normally done directly to buffers located in the job region.

The I/O mapping facility causes the system to "map" I/O transfers through system buffers that are always located in the lower 256K bytes of memory. This facility may be specified selectively for those DMA devices that only have 18-bit controllers or device handlers. The "MAPIO" option for the DEVDEF macro is used to indicate that I/O mapping should be done for a device. Devices which support 22-bit addressing and have device handlers which will execute 22-bit DMA transfers do not need system buffering and can operate normally. See the TSX-Plus Installation Guide for information pertaining to device handlers which support 22-bit DMA.

When I/O mapping is selected for a device, TSX-Plus examines each I/O operation directed to the device and if the buffer is outside of the lower 256K bytes it moves the data from the user's buffer to/from a system buffer and performs the actual data transfer from the system buffer to/from the I/O device. This allows 18-bit devices to be accessed by all time-sharing jobs regardless of

their location in physical memory. However, it introduces a significant speed penalty since the data must be moved between the system buffer and the buffer in the job space. A further speed penalty is introduced in cases in which the amount of data being transferred is larger than the system buffer. In this case, an I/O operation which would normally be accomplished as a single transfer will be broken down into a series of smaller transfers. When a large operation is broken down into a series of smaller operations time is lost waiting for the device to reposition itself for the start of the next operation. This speed penalty can be minimized by allocating a large enough system buffer to accommodate most I/O transfers as a single operation. The generalized data caching facility can also significantly overcome the speed penalty since data read from the cache does not have to be mapped.

3.3 Job scheduling

TSX-Plus schedules jobs for execution based on two factors: (1) the value of a user-assigned job priority that may range from 0 to 127; and (2) the execution state of the job.

3.3.1 Job priorities

Job priority values are arranged in three groups: the fixed-low-priority group consists of priority values from 0 up to the value specified by the PRILOW sysgen parameter; the fixed-high-priority group ranges from the value specified for the PRIHI sysgen parameter up to 127; the middle priority group ranges from (PRILOW+1) to (PRIHI-1). The following diagram illustrates the priority groups:

```
            +-------------+
127     -->|             |
           |   Fixed     |
           |    high     |
           | priorities  |
PRIHI   -->|             |
            +-------------+
           |             |
           |   Normal    |
PRIDEF  -->| interactive |
           | priorities  |
           |             |
            +-------------+
PRILOW  -->|             |
           |   Fixed     |
           |    low      |
           | priorities  |
0       -->|             |
            +-------------+
```

3.3.1.1 Fixed priority jobs: Job scheduling is performed differently for jobs in the fixed-high-priority and fixed-low-priority groups than for jobs with normal interactive priorities. Jobs with priorities in the fixed-low-priority group (0 to PRILOW) and the fixed-high-priority group (PRIHI to 127) execute at fixed priority values. That is, the priority absolutely controls the scheduling of the job for execution relative to other jobs. The job state does not influence the execution scheduling except as to whether the job is in a ready-to-run state or a wait state. A job with a fixed priority is allowed to execute as long as it wishes until a higher priority job becomes active. Jobs having identical fixed priorities are scheduled on a round-robin basis at rates determined by the QUAN0 and QUAN3 parameters.

The fixed-high-priority group is intended for use by real-time programs. See the chapter on real-time program support in the TSX-Plus Reference Manual. The fixed-low-priority group is intended for use by very low priority background tasks. Normal time-sharing jobs should not be assigned priorities in either of the fixed priority groups.

3.3.1.2 Normal priority jobs: The middle group of priorities from (PRILOW+1) to (PRIHI-1) are intended to be used by normal, interactive, time-sharing jobs. Jobs with these assigned priorities are scheduled in a more sophisticated manner than the fixed-priority jobs. In addition to the assigned priority, external events such as terminal input completion, I/O completion, and timer quantum expiration play a role in determining the effective scheduling priority. For these jobs the job state is the primary factor in determining execution scheduling and the user-assigned job priority only influences the scheduling of jobs in the same state.

For most situations, the best strategy is to assign a single priority in the middle of the interactive job priority group to all interactive jobs and reserve the fixed priority groups for real-time or very low priority jobs. The default job priority is specified by the PRIDEF sysgen parameter.

When a job with a normal priority switches to a virtual line, the priority of the disconnected job is reduced by the amount specified by the PRIVIR sysgen parameter. This causes jobs that are not connected to terminals to execute at a lower priority than jobs that are. This priority reduction does not apply to jobs with priorities in the fixed-high-priority group or the fixed-low-priority group. Priority reduction is also constrained so that the priority will never be reduced below the value of (PRILOW+1).

3.3.2 Execution states

TSX-Plus assigns each job a "state" based on actions taken by the job, and external events such as I/O interrupts and timed interval expirations. These states can be grouped into six categories as illustrated by the following diagram:

```
+----------------------+
| Fixed high priority  |     Highest priority
+----------------------+          ^
|     Interactive      |          |
+----------------------+          |
|  Non-interactive     |          |
|  wait completion     |          |
+----------------------+          |
|  Non-interactive     |          |
|   compute bound      |          |
+----------------------+          |
|  Fixed low priority  |          |
+----------------------+          |
|     Wait states      |          v
|  (non-executable)    |     Lowest priority
+----------------------+
```

3.3.2.1 Wait states: Currently, there are sixteen states to identify jobs waiting for events or resources. These jobs are in non-executable states. When a particular event occurs or resource becomes available, the jobs waiting for these events or resources are readily identified by their wait state and are scheduled for execution.

3.3.2.2 Executable states: There are 10 executable job states which can be grouped into five categories: (1) fixed-high-priority; (2) interactive; (3) non-interactive wait completion; (4) non-interactive compute bound; and (5) fixed-low-priority. Jobs that have user-assigned priorities greater than or equal to PRIHI are always in either a wait state or in the fixed-high-priority state. They are never assigned one of the other executable states. Similarly, jobs with user-assigned priorities less than or equal to PRILOW are always in either a wait state or the fixed-low-priority state. Jobs with priorities between (PRILOW+1) and (PRIHI-1) are in one of the states: interactive, non-interactive wait completion, non-interactive compute bound, or wait.

The job scheduler gives preference to interactive jobs to provide rapid terminal response. Each time a job accepts a character from the terminal (this is effectively the same as when a job receives an activation character), the job is classified as "interactive" and the following actions are taken:

1. The job is placed in the highest priority state within the interactive state group.

2. A system timer is started for the job.

3. The I/O count for the job is set to zero.

The job remains in the highest priority interactive state until it either has executed for QUAN1C units of time or performs an I/O operation. At that time, the job is rescheduled into the next lower execution state in the interactive group (interactive-CPU). On return from an I/O operation (during which the job

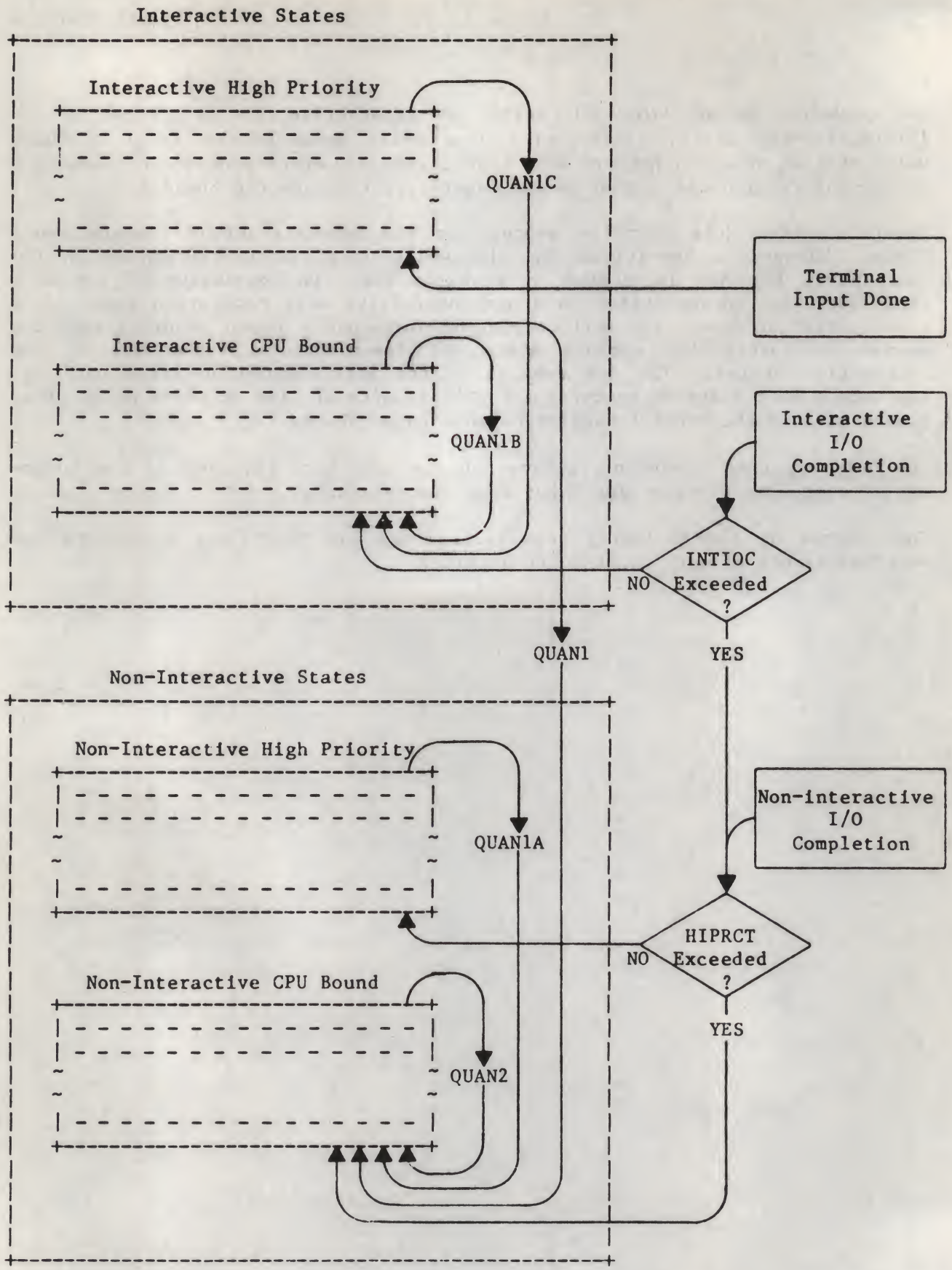
was probably in an I/O wait state) an interactive job is placed in the interactive-CPU state. Interactive jobs which accumulate a total of QUAN1 units of time or which perform more than INTIOC I/O operations are reclassified as non-interactive and placed in the non-interactive compute bound state.

Non-interactive jobs normally execute in the non-interactive compute bound state. Whenever a non-interactive job waits on a resource (such as an I/O operation), the job is placed in a wait state. On completion of the wait condition, the job is placed in a non-interactive wait completion state for a short period of time. The wait completion state has a higher priority than the normal non-interactive compute state but lower priority than any of the interactive states. The job remains in the wait completion state until it reenters a wait state or executes for QUAN1A units of time at which point it is placed back in the non-interactive compute bound state.

The only way that a non-interactive job can move back into one of the interactive states is by receiving input from the terminal.

The diagram on the following page illustrates how time-slice parameters and external events affect job state transitions.

System Overview



3.3.3 Job scheduling algorithm

The job scheduler selects which job to run based on the job states and user-assigned job priorities. The scheduling priority of a job is determined primarily by the priority of the job state and secondarily by the user-assigned priority. In the case of equal state and priority, jobs are scheduled on a first queued - first executed basis. Fixed-high-priority jobs and fixed-low-priority jobs are scheduled solely on the basis of the user-assigned priority value.

The scheduler selects the job to be executed according to the following steps:

1. Select the job in the highest priority state that has the highest user-assigned priority.

2. If this job is not in memory, bypass it and search the job queue in order of decreasing state priority and, within a state, decreasing user-assigned priority looking for a job that is in an executable state and in memory. If there are no jobs in memory in an executable state, then no job is executed until some job enters an executable state or an executable job is swapped into memory.

3. Run the job until: a) the job enters a wait state; b) the allotted time-slice expires; or c) a higher priority job becomes executable.

   a) If the job enters a wait state, remove it from its current queue position and place it in the appropriate wait state queue.

   b) If the allotted time-slice has expired, remove the job from its current queue position and reposition it in the queue based on (1) the priority of its state, and (2) the value of the user-assigned priority. The job is placed behind any other jobs that have the same state and priority. (Note: the quantum expiration may cause the job state to change to a lower-priority state.)

   c) If an external event interrupts an executing job before it either enters a wait state or its time-slice expires, then leave the job in its current state and queue position, but execute the higher priority job. When the interrupted job is resumed, continue its time-slice with the unused remainder of its previous time-slice parameters.

3.4 Job swapping

The role of the job swapper is to keep in memory the highest priority jobs that are in an executable state. A job swap is initiated whenever the swapper determines that there is a job in an executable state out of memory and there is a job in a lower priority state in memory. Note that the wait states have a lower priority than any executable state. When a job swap becomes necessary, the job with the highest priority executable state that is out of memory is selected to be brought into memory. The lowest priority job that is in memory is swapped out of memory to make room for the job being brought in. If this outswap does not yield adequate free memory space, the next lowest priority job is outswapped and the process is repeated until enough space is made available for the selected job to be brought into memory.

The job scheduler attempts to overlap job swapping time with the execution of jobs that are in memory. The "Swapping-I/O" statistic produced by the SYSTAT command indicates the percentage of time that some job swapping was taking place; the "Swap-wait" statistic indicates the percentage of time that no executable job was in memory and swapping was taking place.

A system parameter (CORTIM) is used to keep executable jobs in memory for a reasonable minimum length of time. As long as the job remains executable, it is not eligible to be swapped out of memory until CORTIM units of time (clock time, not the job's execution time) have elapsed. If the job enters a wait state (other than waiting for non-terminal I/O completion), then it becomes immediately eligible for swapping.

Jobs are temporarily locked in memory by the system during non-terminal I/O until released by the device handler in order to make the data transfer into the correct job area. If the job has exceeded its time-slice parameter, then the system "holds" its I/O. Jobs may also lock themselves in memory by using real-time EMT requests.

3.5 Real-time interrupt processing

The real-time interrupt handling facility has two subdivisions: real-time interrupt service routines and real-time interrupt completion routines.

3.5.1 Real-time interrupt service routines

Real-time interrupt service routines provide rapid interrupt response by running at fork level in user mode, but do not maintain the full context of the job. They can execute without requiring a job scheduling cycle, but only a very limited subset of system service calls can be used from within an interrupt service routine. Since real-time interrupt service routines execute at fork level, they are intermixed with system interrupt service fork routines and are queued and executed in order of occurrence. These real-time interrupt service routines will execute prior to any other job regardless of the associated job's priority.

3.5.2 Real-time interrupt completion routines

Real-time interrupt completion routines run with the full context of the job and require a job scheduling cycle before execution. This mechanism does not provide as rapid response as interrupt service routines, but allows normal access to programmed requests from within the completion routine. Each real-time interrupt completion routine has a real-time software priority which is used by the scheduler to compute the execution priority of the real-time completion routine. The priority of the completion routine is calculated by adding the priority specified with the EMT that connects the interrupt to the completion routine to the PRIHI system generation parameter; this priority is constrained by the maximum allowed (127). If a real-time completion routine enters a wait state, then when it resumes execution it returns to the same priority as prior to the wait condition.

A real-time interrupt completion routine may also have a software priority of zero, in which case its execution priority depends on the execution priority of the job. If the execution priority of the job is greater than or equal to PRIHI, then the real-time interrupt completion priority is calculated to be PRIHI and the real-time completion routine is treated the same as above. If the priority of the job is less than PRIHI, then the real-time completion priority is scheduled as a time-shared job in a non-interactive wait completion state.

The following diagram illustrates the processing of an interrupt and shows the relationship between interrupt service routines and real-time interrupt completion routines:

Interrupt Processing

```
+-----------------------------------------------------------------------+
|                                                                       |
|      Level 1                Level 2                Level 3            |
|  -----------------     ---------------        -------------          |
|                                                                       |
|  Hardware interrupt                                                   |
|          |                                                            |
|          V                                                            |
|       .INTEN                                                          |
|          |                                                            |
|          V                                                            |
|        .FORK >-----------------+                                      |
|                                |                                      |
|                                |<--------------+                      |
|                                V               |                      |
|                        +-----------+           |                      |
|                        | Interrupt |           |                      |
|                        |  Service  |           |                      |
|                        |  Routine  |           |                      |
|                        +-----------+           |                      |
|                                |               |                      |
|                                V               |                      |
|                       +--------------+         |                      |
|                       |     More     |         |                      |
|                       |     Fork     | (yes)   |                      |
|                       |   Requests   |-------+                        |
|                       |      ?       |         |                      |
|                       +--------------+                                |
|                         (no) |                                        |
|                              |<-------------------------------+       |
|                              V                                |       |
|                       +--------------+        +------------+  |       |
|                       | Any Pending  |        |  Interrupt |  |       |
|                       | Completion   | (yes)  | Completion |  |       |
|                       |  Routines    |------->|   Routine  |--+       |
|                       |      ?       |        |            |          |
|                       +--------------+        +------------+          |
|                         (no) |                                        |
|                              |                                        |
|          +---------------------+                                      |
|          |                                                            |
|          V                                                            |
|  Return from interrupt                                                |
|                                                                       |
+-----------------------------------------------------------------------+
```

This diagram shows that there are three "levels" of interrupt processing. Level 1 is entered when a hardware interrupt occurs. In this level the processor (hardware) priority is set to 7 which causes other interrupt requests to be temporarily blocked. After some brief interrupt entry processing, the system performs a .FORK operation which queues up a request for processing at fork level and then drops the processor priority to 0. At this time another hardware interrupt can occur, in which case the cycle will be repeated and another request for fork level processing will be placed on the queue.

Level 2 processing is also known as "fork level" processing. This level of interrupt processing services requests that were placed on a queue by the .FORK operation. Hardware interrupts are enabled during this processing and if any other interrupts occur their service requests are placed at the end of the fork request queue. Interrupt service requests are processed serially in the order that the interrupts occurred. Only a limited set of system service calls can be used from service routines running at fork level. One of the valid EMT's is a request to queue a user completion routine for subsequent processing.

Level 3 processing occurs in "job state". That is, the TSX-Plus job execution scheduler selects the highest priority job or completion routine and passes execution to it. During level 3 processing, interrupts are enabled and job execution may be interrupted to process fork level interrupt service routines.

4. DEVICE HANDLERS

TSX-Plus supports the following RT-11 version 5 device handlers: CT, CR, DD, DL, DM, DP, DS, DT, DU, DX, DY, LP, LS, MM, MS, MT, NL, PC, RF, and RK. In addition, the IEEE GPIB version 2 IB device driver is supported. The logical subset disk (LD) and single line editor (SL) are implemented in TSX-Plus as overlay regions and do not require device handlers. The VM.TSX handler is proprietary and unique to TSX-Plus. TSX-Plus also supports a communication line (CL) device handler.

The following RT-11 device handlers are unsupported under TSX-Plus: BA (resident batch handler), EL (error logging pseudohandler), and PD (PDT-11/130/150 handler). In addition, TSX-Plus supports the functionality (but not the RT-11 device handler implementation) for logical subset disks (LD), the single line editor (SL), and the virtual memory device (VM).

4.1 Device handler extensions and restrictions

TSX-Plus requires device handlers which are written to support a memory management RT-11 XM environment. Error logging is not supported under TSX-Plus. See the RT-11 Software Support Manual for details on device handlers. Device handlers must follow the rules for RT-11 XM device handlers in order to function with TSX-Plus.

4.1.1 RT-11 version number checking

TSX-Plus will not install device handlers which were issued by Digital subsequent to the version of RT-11 under which TSX-Plus is being started. In other words, you must upgrade to the appropriate version of RT-11 in order to be able to use the newer device handlers with TSX-Plus. Specifically, the following device handlers require the indicated version of RT-11:

Device	RT-11 version
DU	5.00
XL	5.01

4.1.2 I/O queue element extension

TSX-Plus requires and stores more information concerning each I/O request than does RT-11. To accomplish this, TSX-Plus uses an I/O queue entry which is 14 decimal words long. Each element in the I/O queue has the following format:

Name	Offset	Length	Description
Q.LINK	0	2	Link to next queue entry
Q.CSW	2	2	Address of CSW for channel making request
Q.BLKN	4	2	Physical block number of request
Q.FUNC	6	1	Special function code
Q.UNIT	7	1	Device unit number (bits 0 through 2)
Q.JNUM	7	1	Job number issuing request (bits 3 through 7)
Q.BUFF	10	2	User buffer address relative to Q.PAR
Q.WCNT	12	2	Word count (+ =Read, 0 =Seek, - =Write)
Q.COMP	14	2	Address of completion routine for request
Q.PAR	16	2	PAR relocation bias for buffer address
Q.PA5	20	2	Mapping value for kernel PAR 5
Q.UMRX	22	2	Address of UMR block assigned for I/O
Q.CHAN	24	2	User channel # associated with I/O request
Q.DEVX	26	1	Device index number
Q.FLAG	27	1	Device control flags
Q.UMVB	30	2	UMR base register number
Q.UMPB	32	2	Original value of Q.BUFF when I/O was initiated

TSX-Plus stores the number of the job issuing an I/O request in bit positions 11 through 15 of the fourth word of the queue element. A job number of zero implies the I/O request was initiated from the operating system. User job numbers correspond to the TSX-Plus time-sharing line numbers.

4.1.3 Device handlers use of PARs

Any handler that accesses the user's buffer directly by remapping kernel page address register (PAR) 1 must be altered to use kernel PAR 6. Addresses in the I/O queue entries are automatically adjusted to pass virtual addresses within the PAR 6 region (140000 to 157777). In addition, boundary checking must be altered to correspond to this virtual address region. Any handler using PAR 6 must first issue a .INTEN request.

Kernel page address register 5 is also available for use in device handlers which are not loaded as mapped handlers. If PAR 5 or PAR 6 is used within a handler in an interrupt service routine (after doing an .INTEN) they do not have to be saved since the .INTEN will do this; however, if they are used in a handler other than at interrupt or fork level (e.g., on I/O initiation) they must be saved and restored by the handler.

4.1.4 Extension for the LSI-11 bus

On the LSI-11 bus, only the DL, DU and MS handlers are actually supported with full 22-bit DMA capability, and these devices must have controllers which also support 22-bit addressing and the controllers must be so configured in order to achieve actual 22-bit capability. In order to use any DMA device from a program located above 256K bytes in physical memory, the device and handler must be capable of and configured for 22-bit addressing or the device must be declared to use system I/O mapping in its TSGEN device definition. See the description of the DEVDEF macro (in the TSX-Plus Installation Guide) for more information on 22-bit addressing and system I/O mapping. If a DMA device or

handler does not support or is not configured for 22-bit addressing and does not use system mapping, then attempts to use it will generally result in "Illegal or uninitialized directory" or "Device I/O error" error messages. Serial devices (such as LP, LS, and DX) which do not use direct memory access do not require 22-bit handlers or controllers or system I/O mapping.

4.2 Device handler programmed requests

TSX-Plus supports the standard code expansion for device handler programmed requests implemented in the RT-11 system macro library (SYSMAC.SML) which include .CTIMIO, .DRAST, .DRBEG, .DRBOT, .DRDEF, .DREND, .DRFIN, .DRSET, .DRVTB, .FORK, .INTEN, .MFPS, .MTPS, .SYNCH, and .TIMIO. See the RT-11 System Support Manual for details concerning the usage of these programmed requests. In some cases (as described below), TSX-Plus provides various extensions to these programmed requests.

4.2.1 .FORK requests

In order to understand the processing of fork requests by TSX-Plus, it is helpful to review the concept of interrupt priorities. The PDP-11 family of computers has 8 interrupt priority levels, numbered 0 through 7. The priority of an interrupt is selected by the device requesting the interrupt. The processor (CPU) remembers the current interrupt priority in the processor status word. An interrupt request is held in a pending state and is not allowed to interrupt the processor if the current interrupt priority is equal to or greater than the pending interrupt priority. Priority level 0 is the priority at which the processor runs when no interrupt is being serviced.

Fork processing under TSX-Plus implements a software based interrupt system which effectively operates at an interrupt priority level greater than hardware priority 0 and less than hardware priority 1. Like the hardware interrupt system, TSX-Plus fork requests have priority values that are specified by the software component that is making the fork request. Also like the hardware interrupt system a fork request may interrupt a currently executing fork request of lower priority but may not interrupt a currently executing fork request of equal or greater priority. The fork priority values range from 1 to 127 (decimal); the higher the numerical value, the higher the priority.

Conceptually, fork priorities correspond to interrupt priorities in the range 0.001 through 0.127. Thus, any hardware interrupt request (which has a priority in the range 1 through 7) can interrupt any fork request. Fork requests are queued in order by priority value. If two or more requests have the same priority value, they are queued in the order in which the requests were made.

The standard instructions generated by a .FORK request are:

```
JSR     R5,@$FKPTR
.WORD   fkblk-.
```

Where $FKPTR is a cell containing the address of the system fork routine and fkblk is the address of a four word fork block. Although TSX-Plus does not actually use the four word fork block specified by the request, it does consult the device handler's fork block when necessary to determine the status of the fork entry.

The standard fork call works under both RT-11 and TSX-Plus. When this form of fork call is used under TSX-Plus, the fork request is queued with a fork priority of 50 (decimal).

An alternative fork request call may be used under TSX-Plus to specify a priority for the fork. The form of this call is:

```
JSR     R5,@$FKPTR
.WORD   100000+priority
```

Where "priority" is a fork priority value in the range 1 to 127. The constant 100000 must be added to the priority value to produce a value which TSX-Plus can recognize as a priority rather than the standard RT-11 format above which points to the offset of a user fork block. The assumption is that a handler fork block is highly unlikely to occur at an offset greater than 100000 from its fork request.

The current TSX-Plus fork priority values are defined in the intial part of TSGEN as follows:

Symbol	Value	Description
FP$MAX	127.	Maximum legal fork priority
FP$RT	100.	Real-time interrupts
FP$CKT	70.	50/60 Hz clock interrupt processing
FP$CDI	60.	Terminal character input processing
FP$CDO	55.	Terminal character output processing
FP$DEF	50.	Default fork priority
FP$IOF	50.	I/O complete
FP$IOA	50.	I/O abort entry
FP$PIO	50.	PI output interrupt processing
FP$CK1	30.	0.1 second clock processing
FP$IOS	12.	I/O initiation
FP$MOV	10.	Move data to/from cache buffer

4.2.2 .TIMIO and .CTIMIO requests

Under TSX-Plus it is not necessary for a handler to go to fork level before issuing .TIMIO and .CTIMIO requests. If a job number is placed in the timer control block used with a .TIMIO request, the handler will be synchronized with the specified job number when the timeout routine is entered. If a zero job number is specified in the timer control block, the handler timeout routine will be running at fork level but not synchronized with any job if an I/O timeout occurs. See the RT-11 Software Support Manual for more information on the .TIMIO and .CTIMIO programmed requests.

4.3 Generating device handlers for use under TSX-Plus

TSX-Plus generally uses standard RT-11 XM device handlers, however, some handlers supplied with RT-11 require minor modifications to function correctly with TSX-Plus. The necessary handler modifications have already been applied and are included in the dd.TSX handlers supplied with TSX-Plus.

If you ordinarily need to make no modifications to the handlers supplied by Digital on your system, then you may use the handlers provided with the TSX-Plus distribution. Most common changes can be accommodated through device SET options. However, if you need to change the handlers supplied with RT-11, you may need to apply some patches before using them. See the TSX-Plus Installation Guide for information concerning patching device handlers for use with TSX-Plus.

4.3.1 Building device handlers

When building device handlers, it is necessary to set certain switches before assembly which control conditional code exclusion and inclusion. TSX-Plus requires memory management and optionally allows device timeout. However, it does not support error logging, therefore, error logging should be excluded when the handlers are built.

An easy method of building device handlers for TSX-Plus is to create a TSX-Plus conditional file. Using the editor, create a file "TSXCND.MAC" with the following conditionals:

```
MMG$T     = 1  ;enable memory management
ERL$G     = 0  ;disable error logging
TIM$IT    = 1  ;optionally enable timeout
```

Note that setting a conditional parameter to zero (0) disables the option and setting it to one (1) enables the option. Since device timeout is optionally supported, TIM$IT may be either 0 or 1. Other parameters may be included to specify device characteristics. Refer to Appendix C of the RT-11 System Generation Guide for an entire list, default value, and description of device conditionals. These parameters are not required and will use a default value if left unspecified, except MMG$T which must be set to 1 for TSX-Plus.

Before building device handlers, the appropriate patches (provided as .SLP files on the distribution medium) must be applied. See the TSX-Plus Installation Guide for information on applying the patch files. The distribution medium contains the .SLP files for all device handlers which require modification for use with TSX-Plus. Most handlers may be built by the following commands:

```
MACRO  TSXCND+dd/OBJ
LINK/EXE:SY:dd.TSX  dd
```

where "dd" represents the two character device name.

Only the file structured magtape handlers require different commands. They may be built by using the following commands:

```
MACRO  TSXCND+FSM/OBJ
MACRO  TSXCND+td/OBJ
LINK/EXE:SY:dd.TSX  td,FSM
```

where "td" represents the tape device source module name (TJ, TS, or TM) and "dd" represents the corresponding magtape device name (MM, MS, or MT). Notice that the LINK command automatically appends the "TSX" file extension. Since TSX-Plus uses handlers with the extension "TSX", the handlers must be linked with that extension rather than with the extension "SYS". This allows the TSX-Plus handlers to coexist on the same system disk with standard RT-11 handlers without conflict. Handlers for all devices included in your TSGEN DEVDEF list, including the system disk, must be on the system disk when TSX-Plus is started.

4.3.2 Defining device handler attributes

For each device to be available to the system an entry must be made in TSGEN using the DEVDEF macro. (Note that CL, LD, TT, and SL have other generation parameters and must never be included in a DEVDEF declaration.) This entry requires optional parameters which specify the characteristics of the device handler. Based on these characteristics, TSX-Plus can determine any special operating considerations. The standard device drivers distributed with TSX-Plus have predetermined flag settings known by the initializer. Therefore, it is not necessary to specify flag options when using the device handlers distributed with TSX-Plus. In cases where non-standard handlers are installed, it is necessary to choose the correct device attributes to insure correct operation.

The nine optional device parameters control the following operation:

DMA - Device performs Direct Memory Access (DMA).
In UNIBUS systems with more than 256 K bytes of memory, TSX-Plus allocates and controls UMRs (Unibus Mapping Registers) to perform I/O requests for a DMA device.

MAPIO - Perform I/O mapping.
In QBUS systems with more than 256 K bytes of memory, TSX-Plus must buffer the I/O request into an 18-bit addressable memory region and move the information into the user's area when the users job is above an 18-bit memory address. Requests are not buffered if the job is below the 18-bit low memory region (256 K bytes).

EVNBUF - Require even byte buffer address for I/O transfers.

Some device controllers (DMA devices) and device handlers (VM) which implement a word transfer (rather than byte), require the buffer address to begin on a even byte address (word aligned). In these cases, odd byte addresses may cause I/O failure or fatal system errors which could halt the machine's execution. TSX-Plus will check the buffer address to insure that the transfer is word aligned. If the I/O request does not begin on a word boundary, a user error will be returned from the EMT request.

NOCACHE - Do not use generalized data cache for this device.

For certain devices, it is desirable to disable generalized data cache. For example, since the VM handler uses memory as a device, it would be wasteful of machine resources to also allow it to utilize generalized data cache. This would not only result in displacement of information contained within the cache but would also have the additional overhead of a useless memory to memory transfer.

NOMOUNT - Do not allow mounts for this device.

If this option is specified, the physical device cannot be mounted and therefore will not use directory caching.

REQALC - Require device allocation before use.

If this option is specified, access to the device units is only allowed to users who have allocated the device by use of the ALLOCATE command.

MAPH - Load the device handler into a mapped handler region.

TSX-Plus will place device handlers within an extended memory region, reducing the size of the low memory kernel region (restricted to 40 K bytes). Handlers which are placed in extended memory are known as "mapped" handlers. TSX-Plus communicates with mapped device handlers by mapping PAR5 to the handler's extended memory base address. As device handlers are loaded, the interrupt entry point is intercepted and directed to a low memory address which will map to the handler then enter the handler's interrupt entry code.

Handlers may be mapped under the following conditions:

1. Since only one PAR register is used to access the device handler it must not be larger than 8 K bytes.

2. Since handlers are accessed by kernel PAR 5, the handler must not use kernel PAR 5.

3. Since only two device interrupt vectors per handler are redirected, the handler may not connect to more than two device interrupt vectors. In addition, since the redirection is performed during initialization, the handler may not dynamically connect to interrupt vectors.

4. When the device handler contains an internal buffer used for DMA access, it must calculate the correct physical address taking into account it's own mapped address. It must also declare the HANBUF option which will not allow it to be mapped on extended UNIBUS configuration or when MAPIO is also specified. See the HANBUF option for more information concerning this restriction.

NOMAPH - Always load the handler into the low memory 40k byte region.
Some device handlers are not eligible for mapping into extended memory regions and TSX-Plus will place them in the low memory kernel region. The NOMAPH option will take precedence over the MAPH option if both are specified.

HANBUF - Handler contains an internal I/O buffer used for DMA transfers.
Handlers with internal DMA buffers require special coding to be used as a mapped device handler. In addition, when TSX-Plus is evaluating the system definitions and device characteristics for loading device handlers, it will never map a handler which uses an internal buffer if the handler also requires mapped I/O transfers in QBUS systems with more than 256 K bytes of memory (MAPIO) or if the handler resides in a UNIBUS system with more than 256 K bytes of memory.

The following subroutine illustrates how a handler can translate the virtual address of an internal I/O buffer into the correct physical address. This subroutine will function correctly under TSX-Plus whether or not the handler is mapped:

```
;  Calculate a 22-bit buffer address from the
;  virtual address of a buffer which is contained
;  within the handler.
;  This is necessary if the handler is mapped.
;
;  Input:
;     OLDBA - Virtual address of the internal buffer
;     EXTADR - Zero
;
;  Output:
;      OLDBA - Low order (16-bits) of the physical address
;      EXTADR - High order (6-bits) of the physical address
;      All registers are preserved
;
MAPADR: CMP     OLDBA,#120000    ;Handler loaded into a mapped region?
        BLO     10$              ;Br if handler is below mapped region
        MOV     R0,-(SP)         ;Save registers
        MOV     R1,-(SP)         ;
        MOV     @#172352,R1      ;Get the handler PAR5 relocation value
        CLR     R0               ;Clear high order address cell
        ASHC    #6,R0            ;Convert to physical memory address
        BIC     #160000,OLDBA    ;Isolate offset (clear virtual PAR #)
        ADD     R1,OLDBA         ;Add the phys. relocation to low order
        ADC     R0               ;Add carry to high order
        ASH     #4,R0            ;Shift high order to bits 4 thru 9
        MOV     R0,EXTADR        ;Store high order address
        MOV     (SP)+,R1         ;Restore registers
        MOV     (SP)+,R0         ;
10$:    RETURN                   ;Return
```

4.4 Debugging a device handler

A special version of the ODT debugging program is provided with TSX-Plus for use in debugging user written device handlers. In order to perform this type of debugging the following two files must be available on the system disk when TSX-Plus is started: TSXDB.SAV and SYSODT.REL.

In order to start TSX-Plus under control of the system debugging program, type

```
R TSXDB
```

This is analogous to the R TSX command that is normally used to start TSX-Plus but has the effect of loading the system debugging program into memory with TSX-Plus and transferring control to it before TSX-Plus performs its initialization.

The system debugging program requires approximately 2,900 bytes in the 40K byte low-memory portion of TSX-Plus. If your generated system is so large that there is insufficient space to start it with the debugging program, regenerate the system with all handlers removed except the ones that are needed for execution and reduce the number of time-sharing lines.

When you start the system by typing R TSXDB, it responds by printing an asterisk indicating the debugger is in control and waiting for a command. The debugger only accepts commands from the machine's console terminal. The commands take the same form as for standard ODT except some of the lesser used commands (such as searching through memory) have been removed to save space.

On entry to the debugger, register R1 ($1) contains the address of an instruction that is executed each time control-R is typed. If you set a breakpoint at this location you can trigger a breakpoint whenever you wish, after the system is started, by typing control-R at any active terminal.

Once the control-R breakpoint is set, start the system by typing ";G". After the system has started and you have logged in, determine the address of the base of the handler by use of the SHOW DEVICE keyboard command. The handler being debugged should be specified as non-mapped (i.e., do not load into extended memory) when the system is generated. As a result of this, the "P. base" (physical memory) base address will be zero, and the "V. base" address will be the actual address of the base of the handler. This is the address of the first cell in the handler header (block 1 of the handler) which contains the device vector address. Note, this address is the base of the handler not the handler entry point; the entry point is at base address + 12 (octal).

Once the base address of the handler is known, trigger a breakpoint by typing control-R, set a debugger relocation register at the base of the handler, set breakpoints within the handler, and proceed by typing ";P".

The following sequence of commands illustrates the process of starting TSX-Plus with the system debugger. The values shown for addresses within the system will vary depending on the options included when the system is generated.

```
.R TSXDB          ! Start system with debugger
ODT  V04.00       ! Debugger indicates it is in control
*$1/020504        ! Get address of control-R breakpoint
*020504;1B        ! Set a breakpoint at control-R
*;G               ! Start the system
```

In the case where you are debugging a handler which fails before the system is started (such as the handler for the system device), the procedure for debugging is somewhat more complex. You still start the system under debugger control by typing "R TSXDB", but instead of relying on the control-R breakpoint (which requires that the system get started) set a breakpoint at the location whose address corresponds to the symbol INIJMP which can be found in the TSEXEC section of the system LINK map. The instruction at INIJMP is executed after

the portion of the system initialization that loads handlers but before the system handler is used. Once you have set a breakpoint at INIJMP, start execution by typing ";G".

Once the breakpoint at INIJMP is triggered, you must determine the base address of the handler by examining system tables. The table PNAME (in the TSGEN portion of the map) contains a one word entry for each device; the entry is the RAD50 name of the device. Examine each entry in this table, using the "X" debugger command to convert the octal value to RAD50, until you locate the entry for the handler to be debugged. The HANENT table (also in the TSGEN section of the map) is a table that is parallel to PNAME and contains the addresses of the handler entry points. Examine the entry in HANENT that has the same offset as the device name in PNAME. This is the entry point of the handler. Subtract 12 (octal) to obtain the address of the base of the handler. Breakpoints can then be set in the handler and execution continued by typing ";P".

4.5 Special TSX-Plus device handlers

Several device handlers are uniquely integrated into the TSX-Plus environment. The communication line handler (CL), the logical subset disk handler (LD), the terminal line handler (TT), and the single line editor (SL) are provided as integrated system features and do not require a device declaration (DEVDEF). The professional interface handler (PI) is provided with PRO/TSX-Plus and when used is defined as a shared run-time system and not as a device handler. The TSX-Plus virtual memory handler (VM) utilizes knowledge of the TSX-Plus environment to determine the usable memory space available. It is the only special TSX-Plus device handler which requires a device declaration (DEVDEF) in order to be used.

4.5.1 Communication line handler (CL)

The CL handler allows Input/Output operations to be performed to serial communication lines connected to DL11, DLV11, DZ11, DZV11, DH11, and DHV11 communication controllers. With the CL handler it is possible to have some lines on a multiplexer used as TSX-Plus time-sharing lines, and other lines on the same multiplexer used to drive I/O devices such as printers, plotters, and modems. It is also possible to use a line as a time-sharing line some of the time and as a communications line at other times. Up to 8 communication lines can be controlled by the CL handler. Some of the important features of the CL handler are summarized below:

1. Up to 8 communication lines may be controlled through the CL handler. The device names are "CL0:", "CL1:", ... "CL7:". The lines may be connected to any type of communication controller that is supported by TSX-Plus and may share the same multiplexer controllers as TSX-Plus time-sharing lines.

2. Lines may be dedicated as communication lines or may be switched between time-sharing lines and communication lines.

3. Internal queueing is used within the handler to allow concurrent Input/Output operations to be performed on all of the lines.

4. The CL handler allows both input (read) and output (write) operations. Full duplex (simultaneous) read and write operations may take place on each line.

5. The communication lines may be used with the TSX-Plus spooling system to allow spooled output to devices on communication lines.

6. The CL handler responds to XON/XOFF (control-Q/control-S) control characters to stop and start its transmission and will generate XON/XOFF characters to control the speed of a device transmitting to a CL line.

7. A "binary mode" is available for CL lines to allow full 8-bit, transparent I/O to devices.

8. Modem control is supported. Ring and carrier detect signals may be monitored and data terminal ready (DTR) can be controlled by a program or SET command.

9. The CL handler is implemented as a system virtual overlay, minimizing the amount of code and data that is required in the unmapped portion of the system.

10. The CL handler can be used as a replacement for the LS and XL handlers (the XL handler is used with the RT-11 VTCOM program).

Once a system has been generated with communication lines, the lines may be accessed as normal devices using the names CL0, CL1, etc. If the CL handler is used to drive the system printer, it is convenient to use an assign command to assign the logical name LP to the corresponding CL device.

The device name "CL" is functionally equivalent to "CL0". Attempts to use a CL unit which is not currently associated with a line will return an error status just as if the CL device was not recognized by the system.

4.5.1.1 I/O operations: The .READ/.READC/.READW and .WRITE/.WRITC/.WRITW EMT's may be used to perform standard read/write operations to the CL lines. The CL handler allows full duplex input/output operation, which means that read and write operations may be simultaneously active on a CL line.

When a .READ[C/W] EMT is used to read from a CL line, the operation is complete when the requested number of words have been accepted or a control-Z character is received.

The input character silo is used to store characters received from the line. This buffer prevents characters from being lost during the interval when one read EMT is completed and another is issued to the CL handler.

The CL handler responds to received XON (control-Q) and XOFF (control-S) characters, starting and suspending transmission to synchronize its rate with the device connected to the line.

4.5.1.2 .SPFUN operations: The following special function codes (.SPFUN EMT's) are recognized by the CL handler. The special functions apply to the specific CL unit to which the channel was opened.

Code	Function
201	Clear handler
202	Control break transmission
203	Read with byte count
204	Get handler status
205	Terminate I/O
250	Set option flags
251	Clear option flags
252	Set page length
253	Set skip lines
254	Set page width
255	Get modem status
256	Set line speed
257	Abort pending I/O
260	Read a line of input

Function 201 -- Clear handler. This function clears the internal handler flag that says an XOFF (control-S) character has been received and transmits an XON (control-Q) to the CL device.

Function 202 -- Control break transmission. This function starts or stops the transmission of a break signal. The word count specified with the .SPFUN controls whether transmission of a break signal is started or stopped. If the word count is non-zero, break transmission is started; break transmission continues until another .SPFUN is done with function code 202 and a word count of zero.

Function 203 -- Read with byte count. This function performs a read operation but the "word count" value specifies a byte count instead. This function does not complete until at least one byte is read. However, if a byte count greater than one is specified, bytes are moved from the input ring buffer until either the specified byte count is satisfied or the input ring buffer is emptied. If fewer than the requested number of bytes are available, the remainder of the buffer is filled with nulls. The control-Z character does not signal end of file for this type of read -- control-Z is read as an ordinary character.

Function 204 -- Get handler status. A status code is stored into the first word of the buffer specified with this function. The meaning of the flag bits is as specified below:

bit 0: 1 ==> XOFF has been sent to stop transmission.
bit 1: 1 ==> XOFF has been received from the remote device.

Function 205 -- Terminate I/O to the line. This function "turns off" a communication line. The input ring buffer is emptied (its contents are discarded) and a flag is set causing any other characters received from the line to be discarded. Data Terminal Ready (DTR) status is dropped. The line will be turned on again whenever another I/O operation is performed to it.

Functions 250 and 251 -- Control option flags. These special functions are used to set and clear handler option flags. Function 250 sets the specified flag bits, function 251 clears the specified flag bits. The flag bits correspond to handler SET options. If the option flag is cleared (0), this corresponds to the NOoption setting. The bit positions of the options are shown in the following table. The option flags are contained in a one word buffer for the .SPFUN. For a detailed description of the SET commands, see the TSX-Plus Reference Manual.

Bit	Mask	Option	Function summary
0	000001	FORM	Device supports form feeds
1	000002	TAB	Device supports tabs
2	000004	LC	Send lower case characters
3	000010	LFOUT	Transmit line feed characters
4	000020	LFIN	Accept line feed characters
5	000040	FORMO	Send form feed on block 0 write
6	000100	BINOUT	Binary output mode
7	000200	BININ	Binary input mode
8	000400	CR	Send carriage return characters
9	001000	CTRL	Send control characters
10	002000	DTR	Raise Data Terminal Ready
11	004000	EIGHTBIT	Allow 8 bit character transmission

Function 252 -- Set page length. This function performs the operation of the SET CL LENGTH=n command. The .SPFUN must have a one word buffer containing the number of lines per page.

Function 253 -- Set skip lines. This function performs the operation of the SET CL SKIP=n command. The .SPFUN must have a one word buffer containing the number of lines to skip at the bottom of the page.

Function 254 -- Set page width. This function performs the operation of the SET CL WIDTH=n command. The .SPFUN must have a one word buffer containing the line width.

Function 255 -- Get modem status. This function is used to check on the status of a modem connected to a CL line. The modem status is returned into the first word of the buffer specified with the .SPFUN. The flag bits returned are described below:

Bit	Meaning when set
0	Ring indication
1	Carrier is detected
2	Data Terminal Ready is asserted

Function 256 -- Set line speed. This function is used to set the transmit/receive speed for a CL line. This .SPFUN must have a one word buffer containing a speed code. The following baud rates are represented by the indicated speed codes (speed code values are shown in decimal): 50=0, 75=1, 110=2, 134.5=3, 150=4, 300=5, 600=6, 1200=7, 1800=8, 2000=9, 2400=10, 3600=11, 4800=12, 7200=13, 9600=14, 19200=15. This .SPFUN can only be used for lines connected to hardware controllers that support programmable baud rates such as DLV11E, DZ11, DZV11, DH11, DHV11, and the Professional printer and communication ports. A baud rate of 19200 is not supported by some hardware controllers including the DEC DZ11 controller. The DH11 does not support baud rates of 2000, 3600, or 7200; and the DHV11 does not support baud rates of 3600 or 7200.

Function 257 -- Abort pending I/O. Abort all pending read and write operations issued by the job executing the .SPFUN on the CL unit.

Function 260 -- Read a line of input. This special function reads a line of input terminated by a carriage return character. The "word count" value specified with this special function is interpreted as a byte count. The read terminates when any of the following conditions is met:

1. A carriage return character is received. The carriage return is stored in the buffer and the remainder of the buffer is null filled.

2. The buffer is filled before a carriage return is received.

3. A control-Z is received.

4.5.1.3 Redirecting CL and time-sharing lines: A system service call (EMT) is available to allow a program to assign a CL unit to a particular line. The form of the EMT is:

```
EMT     375
```

with R0 pointing to an argument block of the following form:

```
.BYTE   0,155
.WORD   CL_unit
.WORD   line_number
```

where "CL_unit" is the CL unit number, and "line_number" is the number of a TSX-Plus time-sharing line or dedicated CL line. If the specified line number is 0 (zero), the CL unit is disassociated from any line. Operator privilege is required to use this EMT.

If an error is detected, the C-flag is set on return and the following error codes are returned:

Code	Meaning
1	User issuing the EMT does not have operator privilege
2	An invalid CL unit number was specified
3	An invalid line number was specified
4	The specified line is already assigned to a CL unit
5	A time-sharing user is logged onto the specified line
6	The specified CL unit is currently busy

CL units specified using the CLDEF macro in TSGEN are initially connected to dedicated CL lines. Note that although these lines are dedicated to use by CL, the CL units which are initially assigned to these lines may be reassigned to other lines. The unallocated CL units declared by use of the CLXTRA parameter in TSGEN are initially not associated with any line. The SET CLn LINE=n keyboard command or the system service call (EMT) can be used to assign any CL unit to any free time-sharing line or free dedicated CL line. Thus it is possible to use a line as a TSX-Plus time-sharing line during certain portions of the day and then assign a CL unit to the line and use it to drive a modem or other device during other portions of the day. Dedicated CL lines use less space than time-sharing lines but may only be accessed as CL units. See the TSX-Plus Reference Manual for a full description of the SET CL command.

The following example commands illustrate how CL unit 1 can be assigned to time-sharing line 2. The logical name "LP" is then assigned to CL1 so that the PRINT command will direct output through CL1. CL1 can be declared to be a spooled device in TSGEN:

```
.SET CL1 LINE=2
.ASSIGN CL1 LP
```

The SHOW CL and SHOW TERMINALS keyboard commands can be used to display information about which CL units are associated with which lines. The SHOW CL command also indicates if a CL unit is spooled and lists the options which are set for the unit. See the TSX-Plus Reference Manual for information concerning the SHOW command.

4.5.1.4 VTCOM/TRANSF support and CL handler: The RT-11 VTCOM/TRANSF file transfer programs may be used to communicate and transfer files between RT-11 and/or TSX-Plus systems.

When VTCOM is used to communicate with another system, the system where the user is located and running VTCOM is known as the "local" system whereas the remote system to which communication is taking place is known as the "host" system. TSX-Plus may be used either as the local system, the host system, or both.

The user at the local system runs the VTCOM program to initiate communication with the host system. The VTCOM program uses the CL handler to connect to a communications line. The CL handler must be set up to drive a DL11, DLV11, DZ11, DZV11, DH11, DHV11, or Professional printer or communication port that is connected either directly or through a modem to the host system.

When TSX-Plus is used as the local system, the IOABT sysgen parameter must be set to 1 to enable handler abort entry code.

If the CL handler is used with the VTCOM program, it is necessary to assign the logical name XL (or XC if on the Professional) to the CL unit controlling the communications line. It is also a good idea to allocate the device so that conflicts with other users will not occur. The NOLFOUT option should be specified for a CL line used with VTCOM. For example, the following commands would be appropriate to direct VTCOM to use line CL0:

```
.SET CL0 NOLFOUT
.ASSIGN CL0 XL (or XC)
.ALLOCATE XL (or XC)
```

When TSX-Plus is used as the host system, the connection from the local system may be made through any TSX-Plus time-sharing line on the host system.

4.5.2 IEEE GPIB handler (IB)

When generating support for the IEEE GPIB handler (IB) changes are required to both TSGEN and the IB device handler. The alteration in TSGEN is necessary since the normal IB supplied subroutines attempt to open the IB device on decimal channel numbers 16, 17, 18, and 19. Increasing the TSGEN parameter NUCHN from 16 to 20 (decimal) allocates additional channels and allows the IB subroutines to execute without changes. In non-standard configurations where multiple devices may be used, it may be necessary to allocate more channels. The allocation of these additional channels requires memory in the low memory 40 K byte region and will produce a larger system than a standard system generation.

A change is also necessary to the IB device handler to alter the mapping register used from PAR1 to PAR6. See the section concerning device handlers use of PARs discussed earlier in this Chapter. The following three changes must be applied to the IB handler:

		old value	new value
PAR1	=	172342	172354
PAR1HI	=	37776	157776
PAR1LO	=	20000	140000

4.5.3 Virtual memory handler (VM)

The virtual memory handler (VM) allows memory which is not allocated for use by the operating system to be used as a RAM based pseudo-disk device. The VM handler uses the memory space above the top of memory used by TSX-Plus. TSX-Plus can be limited to using less than all installed memory by specifying the TSGEN MEMSIZ parameter. (See the TSX-Plus Installation Guide for details on the MEMSIZ setting.) Since a memory access is quite a bit faster than a disk access, VM can be use for greater speed in locating and reading files which are frequently accessed.

Since most machines will lose the contents of memory during a power outage, VM should be restricted to read-only, scratch, or executable files. It may be used to speed the execution of heavily overlaid programs or store temporary intermediate sort or work files.

After TSX-Plus is started, VM must be initialized before it can be used. Since VM is implemented as a block structured device, and each block contains 512 bytes, the number of blocks available to VM will be two times the number of K bytes allocated. The directory does require some storage and therefore the number of blocks reported after initialization will be slightly smaller than this total. For instance, in a system which contains 512K bytes total physical memory and with MEMSIZ=256., VM will have 256K bytes available. After initialization, a directory of VM will then show slightly less than 512 blocks.

VM will normally calculate the correct base address to use to be just above the last address used by TSX-Plus. You may increase this base address. The format of the SET command used to adjust the base address used by VM is:

```
SET VM BASE=nnnnnn
```

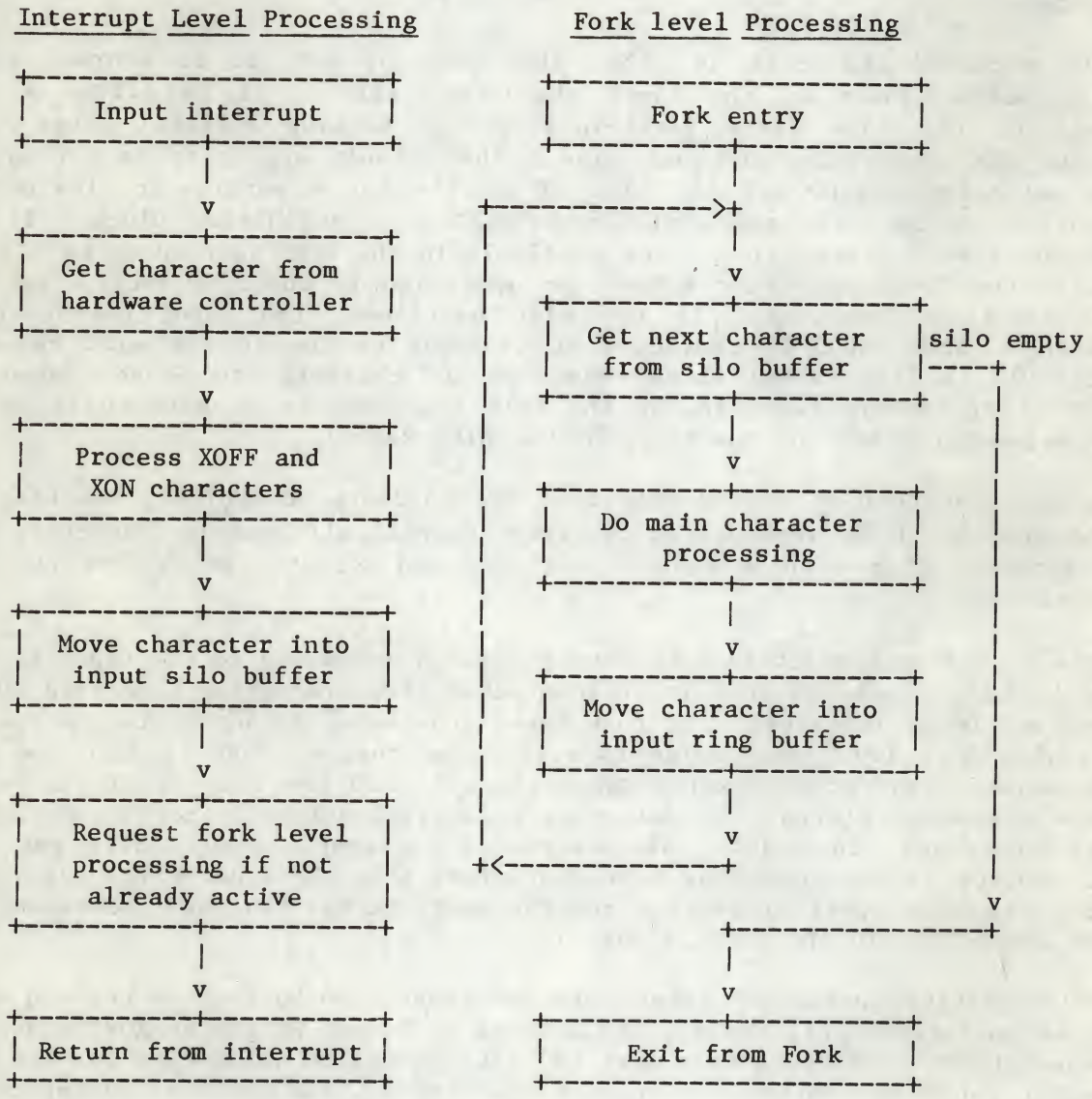
where "nnnnnn" represents bits 6 through 22 of the base memory address (in octal) which VM is allowed to use. However, if you specify a base address below the top address of TSX-Plus, VM will dynamically adjust this base address back above the top of TSX-Plus. For example, if you wish to set the base address of VM to start after the first 512K bytes, then "nnnnnn" should be 20000 since the base memory address is 2000000 (octal). Any time a new base address is defined, VM should be initialized.

5. TERMINAL AND CL INPUT/OUTPUT PROCESSING

5.1 Terminal input character processing

To achieve maximum efficiency, TSX-Plus divides the terminal character input processing into three sections. The first section performs minimal character processing and runs at device interrupt priority level. The second section performs more lengthy character processing and runs at fork level. The third level passes characters to requesting programs and runs at program level. This system was chosen to minimize both system overhead and the time spent running at interrupt level. The interrupt and fork level character processing routines are illustrated in the following diagram:

Terminal Input Character Processing



5.1.1 Interrupt level input character processing

In order to minimize system overhead and the length of time running at interrupt level, only a small amount of character processing takes place in the input character interrupt routine. When an input interrupt occurs, the input interrupt routine is entered at device interrupt priority level. The received character is checked to see if it is an XOFF (control-S). If so, a flag is set causing character output on the line to be suspended. If the terminal controller is a DMA device such as a DH-11 or DHV-11 the current DMA transfer is aborted and information is saved to allow it to be restarted later. Next the character is checked to see if it is an XON (control-Q). If so, the output-suspension flag is cleared for the line and transmission to the line is restarted.

If the received character is other than XOFF or XON, it is stored into a holding buffer known as the input character "silo". It is called a silo because it functions as a first-in-first-out holding buffer. There is a separate character silo for each line. The default silo size is set by the TSGEN parameter NCSILO and the size of a silo for a particular line may be controlled by the SILO macro within a TSGEN line definition block. If the number of free character positions available in the silo is reduced to a value equal to the TSGEN parameter NCXOFF (or specified by the SILO macro), an XOFF character is transmitted. If the silo overflows, the input character is discarded. When an XOFF character is transmitted due to the silo becoming nearly full, a flag is set which causes an XON character to be sent when the number of characters remaining in the silo decreases to a value equal to the TSGEN parameter NCXON (or specified by the SILO macro).

The process of getting a character from the hardware controller, checking it, and storing it in the input silo, is repeated until all pending characters have been accepted from devices such as DZ(V)-11 and DH(V)-11 which have hardware silo buffers.

After all pending characters have been processed and moved to the input silo, a check is made to see if fork level input character processing is active due to a previous input interrupt. If fork level processing is not active, a flag is set saying fork level processing is active and then a .FORK is done and fork level input character processing takes place. Once the fork level processing routine has been entered, the interrupt priority level is 0 (zero) and further input interrupts can occur. If additional interrupts occur while the fork level routine is running, they move characters into the input silos but do not reenter the fork level processing routine until it has finished processing all of the characters in the input silos.

The most critical parameter related to the input silo buffers is the one which controls how nearly full the silo is allowed to become before an XOFF character is transmitted. This parameter must be large enough to allow time for the XOFF character to be transmitted by the TSX-Plus system, and received and processed by the remote system. The minimum acceptable value depends on the speed of transmission and the responsiveness of the remote system. The recommended value is 12.

The silo parameter that controls when an XON character is transmitted is not critical. The recommended value is 4.

The total silo buffer size should be large enough to allow some room between the XOFF and XON points. If the character input rate is slow (for example from a terminal being driven by a typist) the buffer size can be as small as a few characters plus the XOFF and XON parameter values. However, if the input is being received from another computer which can send high speed bursts of characters, then the buffer size should be increased to avoid rapid transmission of XOFF/XON character pairs. The ideal size of the silo buffer for this type of application is equal to the received packet size plus the XOFF cutoff parameter value. The silo buffers occupy space in the 40Kb low memory portion of TSX-Plus so they should not be made excessively large.

5.1.2 Fork level input character processing

The work performed on each character in the fork level routine is much more extensive than that done at interrupt level. The major tasks performed in the fork level routine are summarized below:

1. Processing of control characters such as control-C, control-W, control-U, control-R, and delete.

2. Checking for activation characters (such as carriage return) and restarting a program waiting for terminal input when one is received.

3. Echoing characters.

4. Checking for field width activation and field width limits.

5. Storing the character into an input ring buffer.

In addition to the input silo buffer, each line has an "input ring buffer" which is used by the fork level routine to hold characters until they are accepted by the program. The default size of the input ring buffer is set by the DINSPC sysgen parameter but it may be controlled on a line-by-line basis by use of the BUFSIZ macro in TSGEN. If the number of free character positions in the input ring buffer is reduced to 8, a flag is set for the line preventing further characters from being moved out of the silo buffer. Thus characters will accumulate in the silo buffer until it is nearly full at which time an XOFF character is transmitted.

The fork level routine executes until all pending characters in silo buffers for all lines have been processed.

5.1.3 Program level input character processing

The third level of character processing takes place at program execution level. The primary purpose of this routine is to move characters from the input ring buffer to the user program as they are requested by system service calls such as .TTYIN, .GTLIN, and .READ. If the single line editor (SL) facility is active, its processing is performed within this routine. This routine is also responsible for suspending the execution of a job which requests terminal input when no more activation characters have been received.

5.2 CL input character processing

Input character processing for CL lines is organized in a fashion similar to that used for time-sharing lines. The same input interrupt routine is used for time-sharing lines and CL lines and the same silo buffers are used to store characters as they are received. However, different routines are called to perform fork level processing. Fork level processing for CL lines is simpler than for time-sharing lines since there are fewer significant control characters. Also, no input ring buffers are used for CL lines. Instead, the fork level routine moves characters from the silo buffer into the data buffer specified for the .READ issued to the CL device.

5.3 Terminal output character processing

As with terminal input character processing, terminal output character processing is separated into levels. However, there are only two levels for character output processing: program level and interrupt level. The diagram below illustrates the output character processing routines:

Terminal Output Character Processing

```
Interrupt Level Processing          Program Level Processing

+-----------------------+           +-----------------------+
|   Output Interrupt    |           |    Main character     |
+-----------+-----------+           |      processing       |
            |                       +-----------+-----------+
            v                                   |
+-----------+-----------+                       v
| Get next character    |           +-----------+-----------+
|  from output ring     |           | Place character in    |
|       buffer          |           | output ring buffer    |
+-----------+-----------+           +-----------+-----------+
            |                                   |
            v                                   v
+-----------+-----------+           +-----------+-----------+
| Transmit character    |           | Start transmitter     |
+-----------+-----------+           +-----------+-----------+
            |                                   |
            v                                   v
+-----------+-----------+           +-----------+-----------+
| Return from interrupt |           |        Finished       |
+-----------------------+           +-----------------------+
```

5.3.1 Program level output character processing

Almost all of the processing done on characters being transmitted is performed at program level. The major character processing operations are summarized below:

1. Check for TSX-Plus terminal control program operations such as defining a new activation character, setting field width, etc. (Lead-in character followed by function code.)

2. Optionally perform terminal logging.

3. Optionally expand tabs into spaces.

4. Optionally convert form feed characters into 8 line feeds.

5. Place character in output ring buffer.

As each character is processed by the program level routine, it is checked to see if it is a control character which requires special processing such as tab, form feed, and the TSX-Plus lead-in character. Characters are then stored into the output ring buffer for the line. The default size of output ring buffers is set by the DOTSPC sysgen parameter or may be set for an individual line by use of the BUFSIZ macro in TSGEN. If the output ring buffer becomes full, the execution of the job is suspended until the number of characters in the output ring buffer equals the OTRASZ sysgen parameter, at which point the execution of the job is resumed. As characters are placed in the output ring buffer, a routine is called to try to start transmission to the line.

In the case of communication controllers such as DH-11 and DHV-11 which support DMA transmission, an additional routine is called to move characters from the output ring buffer into linear buffers that are used for the DMA transmission.

5.3.2 Interrupt level output character processing

Since most of the character processing is done at program level, the output interrupt level routine is very simple. It removes the next character from the output ring buffer and, if a character is available, transmits the character. It also checks an output-suspended flag which is set when an XOFF character is received.

5.4 CL output character processing

Output character processing for CL lines is similar to that for time-sharing lines. The primary difference is that a separate output ring buffer is used for CL lines than for time-sharing lines. The default size for the CL output ring buffers may be set by use of the CLORSZ sysgen parameter or by use of the BUFSIZ macro in TSGEN. The recommended size for output ring buffers is ((3*baud_rate)/1000+2). For example, the recommended ring buffer size for a 9600 baud line is ((3*9600)/1000+2) or 31.

5.5 Modem control

TSX-Plus provides full support for dial-up lines connected to modems. A line is declared to be a dial-up line by use of the "$PHONE" flag in the line definition block in TSGEN. The SET TT n PHONE keyboard command can also be used. A line which is declared to be a phone line may also be used with a directly connected terminal.

The following RS232 control signals are significant for phone lines:

Signal name	Pin number
Transmit	2
Receive	3
Signal ground	7
Carrier detect	8
Data terminal ready	20
Ring	22

TSX-Plus holds the data terminal ready signal low when a line is not logged on. This causes the modem to keep the phone hung up. When a ring signal is detected, TSX-Plus asserts the data terminal ready signal which causes the phone to be answered. A timer is started when the data terminal ready signal is asserted. If someone does not log on before a time interval equal to the OFFTIM sysgen parameter elapses, TSX-Plus drops the data terminal ready signal, causing the phone to be hung up.

When a user logs off of a phone line (by typing the OFF command or any other method), a timer is started. If the user does not log back on before OFFTIM units of time have elapsed, the phone is hung up.

In order to provide modem control and also allow the same line to be used with a hardwired terminal, TSX-Plus checks the status of the carrier detect signal when the line is initialized (when carriage return is received while the line is logged off). If carrier detect is asserted (high), TSX-Plus assumes that a modem is connected to the line and continues to monitor the carrier detect signal. If carrier is lost for longer than the value specified by the TIMOUT sysgen parameter, it assumes a line disconnection has occurred and logs off the job and drops the data terminal ready signal. If the carrier signal is not asserted during system initialization, TSX-Plus assumes the line is connected to a hardwired terminal and does not monitor the carrier detect signal until the job logs off.

6. SYSTEM TUNING

Since every computer site is unique, there is no single optimum set of parameters for TSX-Plus system generation. Performance depends on both the system configuration and its actual use. It is necessary to analyze the hardware available as well as the type of application programs which are most commonly run. Together with a knowledge of those programs' characteristics and a basic understanding of the performance features of the operating system, decisions can be made to improve system performance. System tuning is an on-going process which becomes more apparent with increased experience.

Since the basic function of the operating system is to execute user programs, the most important tool in tuning system performance is knowledge of the features used by these jobs and the resources available to them.

Above all, have reasonable expectations; do not expect a LSI-11/23 with slow disk devices to perform like a LSI-11/73 or PDP-11/44 with high-speed disks.

Three distinct system concepts interact with each other to affect system performance tuning: memory utilization, job execution scheduling, and I/O optimization.

6.1 Memory utilization

With the drastic reduction in memory price that has taken place in the past few years, and the availability of models of the PDP-11 family such as the 11/23-Plus, 11/24, 11/44, and 11/73 which can address up to 4M bytes of memory, there is a tremendous disparity between the sizes of systems running TSX-Plus. Fortunately, TSX-Plus has the flexibility to run well in small systems and also take full advantage of large memory systems.

The tuning of TSX-Plus is quite a bit different depending on the amount of memory available. From the point of view of system tuning and operation, a "small" system is one which has inadequate memory to simultaneously accommodate all of the time-sharing jobs that are routinely active. A "large" system is one which has more than enough memory to accommodate all active jobs. A "medium" size system is one which has enough memory to accommodate most active jobs and which has some, but not heavy, job swapping. It must be realized that the most careful tuning of a small system will not yield the performance improvement that could be gained from upgrading the system by adding more memory.

In tuning a small system, the primary consideration is to minimize job swapping. This, in turn, reduces to a problem of minimizing the size of the operating system and the amount of space used by frequently run applications.

6.1.1 System memory utilization

The memory utilization of the operating system is discussed and illustrated in Chapter 3. The components over which you have some control are:

1. Optional features such as the single line editor, generalized data caching, PLAS support, real-time support, etc.

2. The number of device handlers.

3. Space allocated for job tables; this depends primarily on the number of lines generated into the system (real, virtual, and detached) and, to a lesser extent, on the size of the terminal character buffers allocated for each line.

4. The use of shared run-time systems that allow multiple users to access a common run-time system.

Since the operating system is permanently resident, keep it to a minimum size. Do not include unused device handlers or unused time-sharing line definitions. Do not include more virtual lines or detached jobs than will be actually used. Do not include optional features which will not be used (e.g. PLAS, performance monitoring, real-time support). The TSX-Plus Installation Guide provides specific information about the amount of memory space used by each optional feature and each additional time-sharing line. Features such as the single line editor, PLAS support, and the generalized data cache are not recommended for small systems.

Include a shared run-time if it will be used regularly by more than two jobs. But, remember that shared run-times are permanently resident and are wasting system space when not in use.

Specify reasonable values for system parameters such as terminal I/O and spool buffers. Some experimentation may be necessary to determine what buffer sizes are necessary to achieve satisfactory performance for the job mix in your situation. Balance the use of adjustable system features with the knowledge that excessive job swapping may be caused by overly large system parameter selections.

6.1.2 User program memory utilization

The memory partition allocated to a job under TSX-Plus is dynamic and may change size from time to time. The key to user memory optimization is to set the partition size to the smallest size possible for each program that is run. The amount of memory that is allocated to the job partition can be controlled through three techniques:

1. The MEMORY keyboard command.

2. A TSX-Plus system service call (EMT).

3. The SETSIZ program that can store into a SAV file a value indicating how much memory to allocate for the program when it is run.

The .SETTOP EMT does not change the amount of memory allocated to a job partition. If the partition size is to be changed while a program is executing, a TSX-Plus specific EMT must be used.

The first step in optimizing program memory utilization is to determine how much memory space is actually needed by each application program. This is most easily done by using the MEMORY keyboard command to set the partition size and then attempting to run the application program. By varying the size specified with the MEMORY command you should be able to determine the minimum amount of memory which can be allocated for each program.

Note that most programs either execute or don't depending on whether there is adequate memory available; however, some programs such as the COBOL-Plus run-time system may execute more slowly if there is restricted memory space. Hence, you should not only determine the minimum amount of memory required to run the program but should also note the effect of restricted memory space on the performance of the program.

Once the minimum memory size has been determined for a program, the SETSIZ program can be used to store a value into the SAV file for the program that automatically sets the partition size each time the program is run. A command file named SETSIZ.COM is provided with the TSX-Plus distribution to set appropriate sizes for system utility programs such as PIP, DIR, KED, etc. Note that the SETSIZ.COM file needs to be executed only once - when the TSX-Plus system is installed.

The default partition size (as specified by the DFLMEM sysgen parameter) should be set to a reasonable value.

The SHOW MEMORY command can be used to determine the distribution of memory resources between the system and users.

6.2 Job scheduling optimization

Eight time-slice and two I/O count parameters are used to control job scheduling. The eight time-slice parameters are QUAN0, QUAN1, QUAN1A, QUAN1B, QUAN1C, QUAN2, QUAN3, and CORTIM. The two I/O count parameters are INTIOC and HIPRCT. These parameters are assigned initial values during system generation. Their values can be changed dynamically during the operation of the system by use of a command of the form:

```
SET parameter value
```

where "parameter" is the name of one of the ten parameters. Values for the time-slice parameters are specified in 0.1 second units. Operator command privilege is required to change the value of a system parameter. Note that system tuning parameters (QUAN0, QUAN1, QUAN1A, QUAN1B, QUAN1C, QUAN2, QUAN3, CORTIM, INTIOC and HIPRCT) are global to all users and may not be set on a line-by-line basis.

The SET SIGNAL command can be used to monitor the job state transitions and is very useful for selecting values for job scheduling parameters. The form of the SET SIGNAL command is:

SET SIGNAL [NO]parameter

where "parameter" is one of the following system parameters: QUAN0, QUAN1, QUAN1A, QUAN1B, QUAN1C, QUAN2, QUAN3, INTIOC, or HIPRCT.

When signaling has been set for a system parameter, the bell will be rung at the terminal of the job which set the signal each time a job state transition occurs because the job has reached the specified parameter value. This allows the system manager to observe how often the job changes state based on different parameter values. The SET SIGNAL command operates on a line-by-line basis and affects only the line that issued the command.

Signaling may be turned on for any combination of parameters, but each parameter must be specified by a separate SET SIGNAL command. Signaling for an individual parameter may be turned off by specifying "NO" in front of the parameter name. All parameter signaling may be turned off by use of the following command:

SET SIGNAL OFF

When a job receives an activation character from the terminal it is classified as "interactive" and placed in the highest priority state within the interactive state group. The job remains in this state until QUAN1C units of time have passed at which time the job is reclassified into a lower priority state that is still within the interactive job state group. Jobs in this group are scheduled on a round-robin basis every QUAN1B units of time.

If a job performs more than INTIOC I/O operations or exceeds QUAN1 units of time before it receives another activation character from the terminal, it is classified as non-interactive and is placed in the non-interactive compute bound state. Jobs in this state are scheduled on a round-robin basis every QUAN2 units of time. Whenever a non-interactive job waits on a resource (such as an I/O operation), the job is placed in a wait state. On completion of the wait condition, the job is placed in a non-interactive wait completion state which has a higher priority than the compute bound state but a lower priority than the interactive states. The job is allowed to run in the completion state for QUAN1A units of time after which is is placed back in the non-interactive compute bound state.

In selecting values for these parameters, the following guidelines should be considered: It is highly desirable that interactive jobs such as data entry applications and editing programs be classified as interactive through each terminal interaction. Thus, QUAN1 should be set large enough so that the <u>total</u> CPU time used by the application program during one interaction can be completed. Note that if a job performs I/O operations the CPU time counter is suspended (time is not counted while a job is in a wait state) and restarted

(but not reinitialized) when the I/O operation completes. Also, the INTIOC parameter should be set to a value large enough to allow all I/O operations required during a single interactive transaction to be completed.

It is much better to select values for QUAN1 and INTIOC that are too large rather than too small. If the values are too large they will allow long running (non-interactive) programs to be scheduled as interactive slightly longer than necessary. If they are too small, interactive jobs will be reclassified as non-interactive (and given a lower priority) while they are executing an interactive transaction.

The QUAN1 and INTIOC system parameters are two of the most critical scheduling parameters. Jobs are classified as interactive from the time that a character is received from the terminal until QUAN1 units of CPU time are used or INTIOC I/O operations have been performed. The following procedure can be used to select optimum values for these parameters:

1. Issue the following keyboard commands:

   SET SIGNAL QUAN1
   SET SIGNAL INTIOC

2. Set INTIOC to a large value by use of the following keyboard command:

   SET INTIOC 1000

3. Run an application program whose execution is to be optimized.

4. From a separate terminal, vary the value of QUAN1 by use of the keyboard set command:

   SET QUAN1 value

   For each trial value of QUAN1, enter several transactions to the application program and see if the bell rings at the terminal running the application program. If the bell rings, increase the value of QUAN1 and try again. The optimum value of QUAN1 is slightly larger (add 1 to 5) than the smallest value found which is large enough so that the bell does not ring while processing a transaction.

5. Repeat the process for INTIOC by setting QUAN1 to a large value (e.g., 1000) and varying INTIOC starting with a reasonable value such as 30.

6. Try several values of INTIOC until the smallest value is found which is large enough to keep the bell from ringing while processing a single transaction. The optimum value for INTIOC is slightly larger than this (i.e., add 2 to 10).

7. After the appropriate value for QUAN1 and INTIOC have been determined, the system default values for these parameters may be set by modifying TSGEN and regenerating TSX-Plus.

Note: When performing this type of optimization, choose the most frequent and important type of transactions for the test. Don't worry about longer and less frequent operations such as chaining between separate programs. The performance measurements should be carried out with a variety of application programs. Then use the largest values of QUAN1 and INTIOC found for the various applications as the standard system values. Note that system parameters (QUAN0, QUAN1, QUAN1A, QUAN1B, QUAN1C, QUAN2, QUAN3, CORTIM, INTIOC and HIPRCT) are global to all users and may not be set on a line-by-line basis.

The QUAN1C parameter controls the length of time after each terminal input activation that a job remains at the highest priority interactive state before being dropped down to a lower priority interactive state. The ideal value for QUAN1C is just large enough to allow programs, such as KED, which do single character processing to complete the processing of a single character at the highest priority state. It is not desirable to set QUAN1C large enough to encompass longer editing operations such as cutting and pasting, or moving to the top or bottom of a file.

To select the optimum value of QUAN1C, use the SET SIGNAL QUAN1C command and find that value of QUAN1C which is as small as possible but which does not cause the bell to ring while performing normal text entry to the editor.

The QUAN1B parameter controls the round-robin scheduling of interactive jobs within the same state. Its value is usually not critical but should be in the same range as QUAN1C (typically 1 to 4).

The QUAN2 parameter controls round-robin scheduling of non-interactive, compute bound jobs. In medium to large systems where most programs reside in memory, the value of QUAN2 is not critical and should be set to a reasonably small value in the range 2 to 5. In small systems, the value of QUAN2 should be set large enough to reduce job swapping that could take place when multiple compute bound programs are running. The recommended value for small systems is in the range 10 to 30.

Each time a non-interactive job completes an I/O operation, or finishes waiting on some other resource, the job is given a priority boost. The job remains in the high priority state until either (1) it goes into a wait state again, such as waiting on another I/O operation; or (2) it has executed for QUAN1A units of time, at which time it is rescheduled in the non-interactive compute bound state. The idea is to give the job a chance to start another I/O operation without having to wait its normal turn for service. This allows I/O intensive jobs to keep their I/O active even if there are multiple compute bound jobs also running.

Jobs with the same user-assigned priority in the fixed-high-priority group are scheduled in a round-robin fashion based on the QUAN0 system parameter. If QUAN0 is set to 0 (zero), no round-robin scheduling is done for high-priority jobs. Jobs with the same priority in the fixed-low-priority group are scheduled in a round-robin fashion based on the QUAN3 system parameter. Note that this round-robin scheduling of fixed-priority jobs only pertains to jobs

that have the same assigned priority value. A job with a higher fixed priority is never time-sliced with a job with a lower priority.

The CORTIM system parameter controls how long a job is held in memory after being swapped in from disk. Each time a job is swapped into memory, a timer is started for the job. The job is not eligible to be swapped out of memory until either:

1. The job begins executing and enters a wait state (other than non-terminal I/O).

2. CORTIM units of time have elapsed.

Note that a job is never swapped out of memory just because a certain time interval has elapsed. There must be a higher priority job in an executable state out of memory to force a lower-priority job to be swapped. The CORTIM parameter serves as a "throttle" to control the job swapping rate. Increasing the value of CORTIM decreases the job swapping rate but slows the interactive response time. Jobs with user-assigned priorities equal to or greater than PRIHI override the CORTIM parameter and may force outswapping of lower priority jobs regardless of the length of time they have been in memory.

6.3 User program optimization

The TSX-Plus performance monitor feature allows the execution of some application program to be monitored and a histogram produced showing the percentage of time spent in various regions of the program.

The use of single character input activation should be minimized because of the frequency with which this places programs in the high-priority terminal input complete state. The use of no-wait character input may degrade system performance even more since this can place the program in a high-priority terminal input completed state without having received an input character. If at all possible, terminal input should be buffered and completed with a specific activation character (this is normally a carriage return although other activation characters may be defined).

During buffered input, the job is suspended and may even be swapped to disk to allow other jobs to execute. High efficiency terminal mode can be used to reduce the system overhead by eliminating much of the special character processing associated with terminal I/O.

6.4 I/O optimization

TSX-Plus uses three basic techniques to improve system I/O efficiency: (1) overlapping of job execution with I/O wait; (2) device data caching; and (3) device spooling. It is not obvious, but true, that memory size is one of the key factors in optimizing I/O with TSX-Plus.

6.4.1 I/O wait overlap with computation

One of the benefits of a multi-user operating system like TSX-Plus is that system resource utilization is improved by allowing multiple users to be accessing different system resources concurrently.

Whenever one job enters a wait state, waiting for a resource such as an I/O device to transfer data, the TSX-Plus job scheduler looks for another job that is ready to run. The second job might initiate an I/O operation on a different device or might compute and utilize the CPU while the first job is waiting on the I/O operation to complete. Thus, in an ideal situation, the CPU could be utilized 100% of the time as could all of the I/O devices. Generally, 100% utilization of all resources is neither possible nor desirable but the overall system utilization is typically much higher than for a single user system.

The SYSTAT command provides statistics that indicate the degree of overlap that occurred between job execution and I/O. The "User I/O" statistic is the percent of time that some I/O was being performed; the "I/O wait" statistic is the percent of time that the system is idle because there is no executable job and some I/O is taking place. If 100% I/O overlap took place, the "I/O wait" value would be 0 (zero) because there would always be some job to run whenever I/O was active. You can demonstrate this by running a small "loop" program that will execute continuously while other jobs perform I/O. The RESET command can be used to reset SYSTAT statistic values.

In attempting to optimize overall system utilization, the first factor to consider is the number of programs that can fit in memory. Naturally the more programs that are in memory and ready to run, the better the system utilization will be. Also remember that job swapping has multiple negative effects on system utilization: the job being swapped into or out of memory cannot be executed but the memory space is tied up during the swap and cannot be used by any other job; the I/O device to which job swapping is being done is tied up by the swapping and may block I/O operations by the jobs that are in memory and want to run.

The QUAN1A and HIPRCT parameters affect the amount of overlap that occurs between compute-bound and I/O-bound jobs. A non-interactive job is given a priority boost each time it completes an I/O operation. This is done to increase the amount of overlap that occurs between compute-bound and I/O-bound jobs.

For example, consider a system that has two continuously executing compute bound jobs and one I/O bound job. If the job priority was not boosted on I/O completion, the following cycle would occur:

1. Initiate an I/O operation.

2. Place the I/O job in a wait state, waiting for the I/O operation to complete.

3. Alternately execute the two compute bound jobs while the I/O is taking place.

4. When the I/O completes, place the I/O bound job at the tail of the compute-bound queue.

In step 4, the I/O job is placed at the tail of the compute bound queue which means that it will have to wait until both compute bound jobs have used up their time slices before it is allowed to execute and initiate another I/O operation.

Instead of this, the TSX-Plus job scheduler handles the situation as follows:

1. Initiate an I/O operation.

2. Place the I/O job in a wait state, waiting for the I/O operation to complete.

3. Alternately execute the two compute bound jobs while the I/O is taking place.

4. When the I/O operation completes, place the I/O job in a higher priority state which causes it to interrupt the execution of the current compute bound job.

5. The I/O bound job executes for a short period of time and initiates another I/O operation.

6. Put the I/O bound job back in the I/O wait state.

7. Resume execution of the interrupted compute bound job.

The effect is that the I/O job is able to keep the I/O device busy by "stealing" time from the compute bound jobs when each I/O operation completes. However, if there are several I/O intensive jobs they may tend to steal so much time from the compute bound jobs that the compute bound jobs receive little or no time. The HIPRCT parameter is used to control this. After HIPRCT consecutive priority boosts, the I/O job is scheduled at the tail of the compute bound state queue, which means that it will not be executed until all other jobs in the compute bound queue have executed for their full time slice.

If HIPRCT is set to 0 (zero), jobs are never given a priority boost on I/O completion. The recommended value is in the range 5 to 50. The SET SIGNAL HIPRCT command can be used to monitor how often the HIPRCT parameter cuts off a priority boost.

QUAN1A should be set to a small value which is just long enough for I/O intensive jobs to perform completion processing for one I/O operation and initiate another I/O operation. For example, a data base application might have to follow a linked list through an index file to find a selected record. The QUAN1A parameter should be set large enough to allow the program time to locate the forward link in each index block and initiate the I/O operation to read the next block. The SET SIGNAL QUAN1A command can be used to monitor the effect of varying the value of the QUAN1A parameter. The recommended value for QUAN1A is in the range 1 to 4.

6.4.2 Device spooling

Spooling is a technique which intercepts output to slow devices, like printers, directs the output to a disk file and then services the printer as it becomes ready for more data. This mechanism is transparent to the user job and returns the job to an active status more quickly than if the job actually had to wait for the slow device to complete the transfer.

When the operating system services an I/O queue request, it temporarily locks the job into its current memory position so that the data transfer can be correctly fulfilled according to the information in the I/O queue element. When the output is sent to a slow device, this would prevent job swapping until the last data was accepted by the handler and the transfer request satisfied. If several users need access to the system, this could seriously degrade apparent system performance to those users waiting to be activated. However, when a slow device is spooled, then the output is redirected to the system spool file and the transfer completes at the faster rate of disk I/O, returning control to the job and permitting it to be swapped if necessary. In addition, TSX-Plus will always attempt to double buffer the spooled output request if two or more buffers have been defined.

6.4.3 Caching

Caching is a technique for improving system performance by keeping in memory a "cache" of the most recently accessed blocks of data. Each time a read operation is performed a check is made to see if the requested data block(s) are in the cache. If so, the data is copied from the cache buffer to the receiving program buffer and no actual device I/O is done. Write operations update the data in the cache as well as writing to the I/O device.

Caching speeds up read operations so that they are performed at the speed that the CPU can move data around in memory rather than the speed of an I/O device. Write operations are somewhat slowed down by caching since updating of the cache must be done as well as writing of the data to the I/O device.

TSX-Plus offers three distinct types of information caching: directory caching, generalized data caching, and shared file data caching.

6.4.3.1 Directory caching: When a program opens an existing file on a disk, it is necessary to determine the location of the file by consulting the file directory on the disk. This results in one or more disk I/O operations each time a file is opened. In order to speed this process, TSX-Plus contains a memory resident cache which contains directory information for a selectable number of files. If one or more jobs open the same file several times, then the ability to locate that file's directory information in the directory cache can eliminate many I/O requests and significantly improve system performance.

The system device directory is always cached; directory caching for other devices can be enabled by use of the "MOUNT" keyboard command. See the TSX-Plus Reference Manual for a detailed description of the MOUNT command.

TSX-Plus manages the entries in the directory cache by retaining those most recently used. When no space is available in the cache buffer to add a new directory entry, the least recently accessed entry is discarded and replaced with the new entry. File operations which change the disk directory information (such as .ENTER, .DELETE and .RENAME) are always "written through" the cache, changing both the directory cache entry and the disk directory. This eliminates the speed advantage on these types of operations, but reduces the chances of data corruption.

It is very important to remember to DISMOUNT a disk when changing removable packs on that device. The DISMOUNT command clears all entries from the directory cache for the device. If this is not done the new pack may be corrupted by use of the (incorrect) directory information maintained in the cache for the previous disk pack. The SHOW MOUNTS command identifies which devices are currently eligible for directory caching. Note that all jobs which have MOUNTed a device must either DISMOUNT it or log off before the device's directory entries are cleared from the cache.

6.4.3.2 Shared file data caching: Shared file data caching maintains memory resident copies of data blocks from files which have specifically been declared to use data caching. After a file is opened in the normal manner, a special system service call must be issued to declare that file eligible for data caching. (Data caching is requested by using the TSX-Plus EMT to request shared access to the file, regardless of the protection level selected.)

When a request is issued to read data from that file, a check is made to see if the requested block(s) are currently in the data cache. If the data is in the cache the data is moved from the cache to the user's program with no disk I/O at all. Data blocks are maintained in the cache according to frequency of use. When the data cache is full, the least active block is replaced whenever a new block is read. This replacement algorithm is highly efficient for files with indexed organization, like COBOL-Plus ISAM files. As with directory caching, the data cache is always written through. That is, if the information in a block in the cache is changed, then the disk copy of that block is also updated. If shared file data caching is used at all, it is recommended that at least 8 blocks be allocated for the cache. If a large area is available for a data cache, it is recommended that the generalized data caching facility (described below) be used instead of shared file data caching.

The number of blocks allocated in memory for the shared file data cache is controlled by the NUMDC parameter in TSGEN. One way to determine the best value for this parameter is to generate a system with a large number of cache buffers and then use the SET NUMDC keyboard command to vary the number of buffers used while observing the effect on system performance. The SYSMON program can be used to display statistics about shared file data caching operation.

6.4.3.3 Generalized data caching: Generalized data caching maintains memory resident copies of data blocks from devices which are mounted using the keyboard "MOUNT" command. Each time a read operation is performed, the memory resident cache of data blocks is searched to see if the block(s) requested are already contained in one of the data cache memory buffers. If the block is in the memory cache, it is moved directly from the cache buffer requiring no disk I/O to be performed. If the block(s) are not within the data cache, they are read into the least recently used data cache buffer(s) and then moved to the requesting job. Write operations update the memory cache as well as writing to the device, thus eliminating the possibility of data loss or corruption.

Unlike shared file data caching, generalized data caching applies to all files that are on mounted devices. This means that SAV files for commonly executed programs such as PIP, KED, TSKMON, and application programs will benefit from the cache as well as program overlay segments, and application data files.

To enable generalized data caching, assign a non-zero value to the CACHE parameter in TSGEN. This causes the data caching code to be included in the generated system and controls the number of blocks of memory allocated for data caching buffers. If data caching is not wanted, set the CACHE parameter to 0 (zero).

A SET command is available to dynamically alter the number of blocks of data held in the data cache. The form of this command is:

```
SET CACHE value
```

This command does not alter the amount of space allocated for the data cache (that is directly controlled by the CACHE sysgen parameter), but can be used to cause the system to use less than the full cache area. Operator command privilege is required to use the SET CACHE command. The primary use of this command is to allow the system manager to experiment with different cache sizes to determine the effect on system performance. Once an optimum cache size has been determined, the TSGEN parameter CACHE can be set to this value and the system regenerated. The SHOW CACHE keyboard command can be used to display the current number of blocks currently being used in the data cache.

The effectiveness of the data caching facility increases with the number of blocks allocated for the data cache. In systems with large amounts of memory it is reasonable to allocate several hundred blocks to the data cache. However it is not wise to allocate so much memory space to the data cache that job swapping is significantly increased due to limited memory space for time-sharing users.

The amount of improvement due to data caching also depends on the ratio of the processor (CPU) speed to the speed of the I/O device being cached. The effects of data caching are most pronounced when a fast processor is running with a slow I/O device. Data caching is not recommended for systems which are primarily bound by CPU utilization rather than I/O throughput.

Data caching can have a dramatic effect on the execution of overlayed programs if the cache is large enough to hold the overlay segments. FORTRAN and COBOL-Plus compilation times are typically reduced by 20% to 40% by data caching.

The following table shows typical cache "hit" rates as a function of the cache size (in blocks) for various language processors performing assemblies or compilations:

Cache size versus percent of blocks read from cache while performing assemblies and compilations

Cache Size	MACRO	FORTRAN	F77	COBOL-Plus	DBL	Pascal-2
20	2%	0%	23%	11%	5%	0%
35	3	1	23	21	9	0
50	4	1	23	82	10	5
75	14	2	24	83	25	8
100	36	2	24	84	45	9
150	48	4	27	84	55	90
175	49	51	33	87	84	90
200	50	87	33	87	84	90
250	66	90	34	87	84	90
275	92	92	35	88	84	91
300	92	93	87	88	84	92
400	92	94	94	95	84	92
500	92	97	94	98	84	93

The single job (non-XM) versions of F77 and Pascal-2 were used in making these measurements.

The following statistics for cache hit rates were measured while running a COBOL-Plus program performing 5000 random reads on an indexed organization (ISAM) file containing 44000 records with a 16 byte key.

Cache Size Versus Hit Rate For Reads From ISAM File	
Cache Size	Cache Hit Rates for Random Reads
5	24 %
10	32
15	38
20	46
25	50
30	55
40	60
50	64
60	65
70	67
80	70
90	71
100	72
200	79
300	82
400	83
500	84
1000	85

These statistics were gathered by generating a TSX-Plus system with a 1000 block data cache and then using SYSMON to measure the cache hit rate while varying the effective cache size by use of the "SET CACHE nnn" command. It is recommended that a similar procedure be carried out to determine the optimum cache size for a given application program.

The shared-file data caching facility should be used instead of the generalized data caching facility in the following cases:

1. If the primary goal is to speed up application programs which make heavy use of shared files, and the memory space which can be devoted to data caching is limited (less than 50 blocks), then the shared-file data caching facility is more effective than the generalized data caching facility.

2. If the size of the unmapped portion of the TSX-Plus system is such that code for the generalized data caching facility cannot be added. Note that the shared-file data caching facility does not add any code to the unmapped portion of the system.

If the generalized data caching facility can be used, it is recommended that the shared-file caching facility not be used (it is redundant) and the NUMDC sysgen parameter be set to 0 (zero).

6.4.4 Virtual memory handler (VM)

The virtual memory handler (VM) allows memory which is not allocated for use by the operating system to be used as a RAM based pseudo-disk device. Since a memory access is quite a bit faster than a disk access, VM can be use for greater speed in locating and reading files which are frequently accessed.

Since most machines will lose the contents of memory during a power outage, VM should be restricted to read-only, scratch, or executable files. It may be used to speed the execution of heavily overlaid programs or store temporary intermediate sort or work files.

VM is similar to data caching and the following considerations may help you to decide which is best suited to your application:

1. Data is "written through" the cache to the I/O device that is being cached. Since there is no I/O device associated with the VM handler, no I/O takes place on write operations. This means that is faster to write to VM than to a cached I/O device. This could make VM considerably faster for a "scratch" file that has as many blocks written to it as read.

2. Data written to VM is volatile and will be lost when the system is shut down or halts due to hardware or software malfunction.

3. The amount of space in VM is fixed at sysgen time and an attempt to use more space will result in a no-free-space error return. The number of blocks allocated for caching affects the performance of the cache but not the capacity. As long as there is available space on the I/O device, it is accessible through the cache.

4. Caching is automatic and transparent to application programs. VM requires that program and data files be copied to VM and that application programs open files on VM.

5. Data placed in VM is held there until it is deleted or the system is restarted. Data in the cache is dynamic and may be replaced by data accessed more recently by other jobs. Therefore the speed of access to data in VM is guaranteed whereas the speed of accessing data through the cache depends on whether the data is currently in the cache.

7. SYSMON - DYNAMIC SYSTEM DISPLAY UTILITY

SYSMON is a dynamic interactive utility program used to display information about system activities at a VT2xx, VT1xx, or VT52 type terminal. It is used to assist the system manager in optimizing system resource use and verifying the function of certain operations. It currently provides dynamic screen displays of: CPU and I/O usage; job status; terminal status; message channel status; user time bar chart; CPU time bar chart; directory cache contents; data cache usage; and CL device status.

SYSMON obtains much of its information from tables within TSX-Plus, and thus requires the use of some real-time functions. The real-time facility of TSX-Plus must be included during system generation in order to use SYSMON. Operator privilege is required to run the SYSMON program. If a user without operator privilege tries to run this program, or anyone tries to run it without real-time support generated, he will receive the message:

```
SYSMON-F-Real-time support not specified in TSGEN or user not privileged
```

7.1 Creating and Running SYSMON

SYSMON is automatically created by the command file which links TSX-Plus. Because it depends on global information from TSX-Plus, it must be relinked whenever TSX-Plus is changed. If TSX-Plus is not linked on the system disk, copy the file SYSMON.SAV from the link output device to SY:. Once this is done, you can run SYSMON by typing:

```
R SYSMON
```

If you do not choose to put SYSMON on the system disk, you must use the RUN command with the full device/file specification. Note that many of the displays shown here are slightly narrower than SYSMON produces; this is done to allow the examples to fit on the page.

7.2 SYSMON Menu

```
+-------------------------------------------------------------------------+
|                                                                         |
|                        TSX-Plus SYSMON Utility                          |
|                        25-Jul-84       14:14:06                         |
|                                                                         |
|                                                                         |
|                                                                         |
| Enter selection :                      Sample time :                    |
| (RETURN to exit)                       (RETURN defaults to 5 seconds)   |
|                                                                         |
|                                                                         |
| 1. System status display               6. CPU modes display             |
|                                                                         |
| 2. Job execution status display        7. Directory cache display       |
|                                                                         |
| 3. Terminal status display             8. Shared file cache display     |
|                                                                         |
| 4. Message channels display            9. Generalized data cache display |
|                                                                         |
| 5. User times display                  10. CL device display            |
|                                                                         |
+-------------------------------------------------------------------------+
```

Once you have started SYSMON, you will be prompted from this menu for a display number and the sample rate. The minimum sample time is one second; you may set the sample time as high as you wish. Be aware that on some systems (notably 11/23 based systems) using a small sample time can have a detrimental effect on system response in general. Once you are in a display, press RETURN to return to the menu.

7.3 System status display

```
+-----------------------------------------------------------------------+
|                                                                       |
|                      TSX-Plus SYSMON Utility                          |
|                      25-Jul-84       14:14:06                        |
|                                                                       |
|                                                                       |
|                 *****  System Status Display  *****                   |
|                                                                       |
| Total Uptime      00:35:41.0      Cur       Total        System Parameters |
|                                                                       |
| User Job Time     00:08:11.6     94.1%     22.9%          QUAN0  =    2 |
|                                                                       |
| I/O Wait Time     00:01:24.8      4.8%      3.9%          QUAN1  =   20 |
|                                                                       |
| Swap Wait Time    00:00:00.6      0.0%      0.0%          QUAN1A =    2 |
|                                                                       |
| Idle Time         00:26:04.0      0.0%     73.0%          QUAN1B =    2 |
|                                                                       |
| User I/O Time     00:02:13.2 *    6.7%      6.2%          QUAN1C =    1 |
|                                                                       |
| Swap I/O Time     00:00:00.6 *    0.0%      0.0%          QUAN2  =   10 |
|                                                                       |
| * - Time is overlapped                                                |
|                                                                       |
+-----------------------------------------------------------------------+
```

The system status display provides information on how time is being used in the system and current settings for the dynamically modifiable scheduling parameters.

Three columns of information are presented for the system time usage display. The first is the total time spent in a given activity since the system was booted. (The RESET keyboard command also clears the time counters as if the system had been booted.) The second column is the percentage of total time spent in that activity during the last sample period. The final column is the percentage of time spent in that activity since the system was booted or the keyboard RESET command was last issued.

Seven rows of information are presented. The first is Total Uptime, that is, the amount of time since the system was booted. The User Job time is the time used in computation, that is, actual CPU activity. The I/O Wait time is the amount of time user jobs spend waiting on information from various I/O devices. The Swap Wait time is the time the system spends waiting for a job swap to complete. The Idle Time is the amount of time the system spends idle. The User I/O time is the amount of time user jobs spend performing I/O. Finally, the Swap I/O time is the amount of time the operating system spends swapping user jobs in and out of memory. Note that the percentages will not always add up to 100 percent. TSX-Plus overlaps I/O and execution time, so I/O time might

be building up on one job as another job is executing. Also, program execution and particularly, the actual sample rate, are dependent on current system load and number of real-time interrupts taking place. As an example, the display may be interrupted while computing I/O wait time, during which the times for the subsequent display items may change.

Job scheduling parameters are displayed down the right side of the screen. The values of these parameters can be dynamically changed during system operation by use of the keyboard SET command.

7.4 Job execution status display

```
+-------------------------------------------------------------------------------+
|                                                                               |
|                        TSX-Plus SYSMON Utility                                |
|                        25-Jul-84       14:14:06                               |
|                                                                               |
|                                                                               |
|                 *****  Job Execution Status Display  *****                    |
|                                                                               |
|  Job   Line   Pri  Program  User Name      Run    Size   State                |
|  ---   ------ ---  -------  ---------      ---    ----   -----                |
|   1    1(0)    50  KMON     TSX                    32     Wait-TT input       |
|   3    3(0)    50  KMON     CRAIG                  32     Wait-TT input       |
|   4    4(0)    50  SYSMON   SAM             I      20     TT input done       |
|   6    6(0)    50  KMON     DIRECTOR               32     Wait-TT input       |
|   7    7(0)    50  KED      SOFTWARE               34     Wait-TT input       |
|   9    Det.    50  RTSORT   [000,000]             60     Wait-message        |
|  11    4(1)    40  KED      SAM                    34     Wait-TT input       |
|  12    4(2)    40  MACRO    SAM             C      60     CPU bound job       |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
+-------------------------------------------------------------------------------+
```

The job execution status display provides a variety of useful information. For each job on the system, it provides:

1. Job Number - the TSX-Plus job number assigned to that particular primary line, virtual line, or detached job.

2. Line - This entry tells the primary line number for the job and the virtual line number of this job for that primary line. If the virtual line number is 0, than the job is on the primary line. If Det. appears, then the job is detached, and as such belongs to no line.

3. Priority - the current priority for this job.

4. Program being run - the name of the program file being executed.

5. User - the name of the user who owns the job. If there is no user name associated with the job (detached job or no name assigned to an account), then the job's project, programmer number will be displayed.

6. Running state - All jobs that are not in a wait state are classified as either interactive or CPU-bound, depending on the nature of the work being done.

7. Memory size - the amount of memory that the job is using - expressed in Kb (1024 bytes).

8. Current execution state - What the job is currently doing. The execution states are:

State Displayed	Meaning
Real time state	Executing high priority real-time job
TT input - sing char act	Input character just received while in single character activation mode.
TT input done	Activation character just received
TT output buffer empty	Terminal output buffer empty
Interactive Job compute	Interactive priority job executing
Timed wait completion	Finished .TWAIT or .MRKT request
TT output buffer low	Ready for program to continue output
I/O completion	I/O transfer completed
CPU bound job	Normal job executing
Low Priority Computation	Low priority job executing
Wait-I/O queue element	Waiting on I/O queue element
Wait-Mapped I/O Buffer	Waiting for buffer to be available
Wait-Cache Control Block	Locating device cache control block
Wait-USR data access	Waiting on access to USR data base
Wait-I/O completion	Waiting for I/O operation to complete
Wait-TT output buf full	Terminal output buffer full
Wait-locked block	Shared file block needed by program lock
Wait-system message buf	Waiting for free system buffer
Wait-spool file space	Print spool file currently full
Wait-TT input	Waiting for activation character
Wait-SPD access	Waiting for access to device data base
Wait-spool entry	Waiting for spool file control block
Wait-message	Waiting for a message
Wait-.SPND/.RSUM	Waiting for .RSUM request after .SPND
Wait-timed interval	Waiting for interval to finish
Wait-memory expansion	Waiting for memory expansion to complete

7.5 Terminal status display

```
+-----------------------------------------------------------------------------+
|                                                                             |
|                          TSX-Plus SYSMON Utility                            |
|                          27-Jul-84       18:03:32                          |
|                                                                             |
|                   *****  Terminal Status Display  *****                     |
|                                                                             |
| Terminal line number =  3                                                   |
| Terminal type = VT100, logged on                                            |
| Single line editor status = Enabled, KED, NOTTYIN, Inactive                 |
| Terminal characteristics = DEFER ECHO NO8BIT NOFORM NOFORMO NOGAG LC PAGE   |
| PRIV NOQUIET SCOPE NOSINGLE TAB NOTAPE WAIT                                 |
|                                                                             |
| Rubout filler character .  " "        Escape-letter activation   disabled   |
| Virtual line . . . . . .  enabled     Transparent output . . .   disabled   |
| Command file input . . .  disabled    Field width activate . .   0          |
| High-efficiency mode . .  disabled    Field limit activate . .   0          |
| Echo LF after CR . . . .  enabled     Default terminal editor .  KED        |
| Single char activation .  disabled    Allow non-wait .TTYIN . .  disabled   |
| UCL setting . . . . . .   middle      Speed . . . . . . . . . .  9600       |
| XOFF received . . . . . .  no         XOFF sent . . . . . . . .  no         |
| Activation characters . .  none                                             |
| Input buffer contents . . <>                                                |
|                                                                             |
|                                                                             |
|                                                                             |
+-----------------------------------------------------------------------------+
```

The terminal status display shows the parameters that are currently set on a given terminal line, various terminal characteristics, the terminal's owner, the type of job the terminal is connected to, and if an ASCII DC3 (control-S) has been transmitted or received. This display will prompt you first for a job number - this can be obtained from the TSX-Plus SYSTAT command or the job execution status display. If you simply type return at this point, you will see the information on your own job. If you enter 0, you will get a cycling display through all of the jobs. For information on the parameters displayed, see the TSX-Plus Reference Manual, Chapters 2, 4, and 6.

7.6 Message channel display

```
+-----------------------------------------------------------------------------+
|                                                                             |
|                          TSX-Plus SYSMON Utility                            |
|                          10-Aug-84       14:55:13                          |
|                                                                             |
|                     *****  Message Channels Display  *****                  |
|                                                                             |
| Job  Channel  Message                                                       |
| ---  -------  -------                                                       |
| 15   RTSORT   SYS:SLSANL.DDF/IT:DRA/SI:250/EOF:134/KEY:CA2.2:DA58.2:DA60.3:|
|               CA222.2:CA224.2:CA16.10:DA91.2:DA87.4                         |
| 15   RTSORT   SYS:GHGARG.DDF/IT:DRA/SI:250/EOF:134/KEY:CA222.2:CA224.2:DA87|
|               .4:CA16.10                                                    |
| 13   TSTMSG   [Message request - completion routine]                        |
|  3   TGTPRG   [Message request - job waiting]                               |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|     MAXMC =   5       MSCHRS = 200      MAXMSG =   6      MAXMRB =  10    |
|                                                                             |
+-----------------------------------------------------------------------------+
```

The message queue display shows waiting messages and their respective channel names. This information is useful to verify correct message channel usage. In this example, a sort command string is sent on the RTSORT message channel without RTSORT running to verify that the proper information is being sent (the message will remain in the channel until read). Similarly, you can verify that a job is waiting on the proper channel, or that a completion request is active. For more information on message channels see the TSX-Plus Reference Manual.

7.7 User times display

```
+-----------------------------------------------------------------------------+
|                                                                             |
|                        TSX-Plus SYSMON Utility                              |
|                        25-Jul-84      14:14:06                              |
|                                                                             |
|                   *****  User Times Display *****                           |
|                                                                             |
|                                   0         25        50        75       100|
|                                   + - - - - + - - - - + - - - - + - - - - + |
| 4 [002,002] TSX        KED     44  %%%%%%%%%%%%%%%%                         |
|                                   |         |         |         |         | |
|11 [006,112] SAM        DPS     49  %%%%%%%%%%%%%%%%%%%%                     |
|                                   |         |         |         |         | |
|12 [006,012] SAM        SYSMON   1                                           |
|                                   |         |         |         |         | |
|             Wait Time           4  %                                        |
|                                   |         |         |         |         | |
|             Idle Time           4  %                                        |
|                                   |         |         |         |         | |
|                                   |         |         |         |         | |
|                                   |         |         |         |         | |
|                                   + - - - - + - - - - + - - - - + - - - - + |
|                                                                             |
+-----------------------------------------------------------------------------+
```

The user times display shows, for each job that used at least one percent of the CPU time during the last sample period: the job number, project-programmer number, user name, program running, percentage of time used by that job during the last sample period, and a bar graph depiction of that percentage. If a job uses less than one percent of the time during the period, it is not displayed for that period. This display is useful for determining how CPU time is being used, and the interrelationship between job time, wait time (which includes both I/O wait and swap wait time), and idle time.

7.8 CPU modes display

```
+-----------------------------------------------------------------------------+
|                                                                             |
|                             TSX-Plus SYSMON Utility                         |
|                             25-Jul-84       14:14:06                        |
|                                                                             |
|                         *****  CPU Modes Display *****                      |
|                                                                             |
|                                                                             |
|                                 0         25        50        75       100   |
|                                 + - - - - + - - - - + - - - - + - - - - +   |
|       User Job Time         90   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%    |
|                                 |         |         |         |         |   |
|       I/O wait Time         10   %%%%                                    |
|                                 |         |         |         |         |   |
|       Swap Wait Time         0                                              |
|                                 |         |         |         |         |   |
|       Idle Time              0                                              |
|                                 |         |         |         |         |   |
|       User I/O Time  *      30   %%%%%%%%%%%%                            |
|                                 |         |         |         |         |   |
|       Swap I/O Time  *       0                                              |
|                                 + - - - - + - - - - + - - - - + - - - - +   |
|                                                                             |
|       * - Time is Overlapped                                                |
|                                                                             |
+-----------------------------------------------------------------------------+
```

The CPU modes display shows the same information about sample period time usage that is shown in the system status display, however, it is shown here in a bar chart format.

7.9 Directory cache display

```
+-----------------------------------------------------------------------------+
|                                                                             |
|                        TSX-Plus SYSMON Utility                              |
|                        25-Jul-84       14:14:06                             |
|                                                                             |
|                   *****  Directory Cache Display *****                      |
|                                                                             |
|                                   Page 1                                    |
|                                                                             |
|   DL0:TSKMON.SAV                          RK0:[UTIL.WORKIT]OUT.DSK          |
|   RK0:[SYSMON]SYSMON.SAV                  RK0:[UTIL]WORKIT.DSK              |
|   RK0:[UTIL..OUT]SUPER.TXT                RK0:UTIL.DSK                      |
|   DL0:KED.SAV                             RK0:SYSMON.DSK                    |
|   DL0:DIR.SAV                             RK0:SYSMGR.DSK                    |
|   DL0:PIP.SAV                             RK0:COBOL.DSK                     |
|   DL1:TSKMN1.OBJ                          DL1:TSKMN3.OBJ                    |
|   DL1:TSKMN2.MAC                          DL1:TSKMN3.MAC                    |
|   DL0:TSXUCL.DAT                          DL1:TSKMN3.BAK                    |
|   DL0:TSXUC..SAV                          RK0:[SYSMON]SYSMON.COM            |
|   DL0:SYSMAC.SML                          DL0:STDASN.COM                    |
|   DL1:TSKMN1.MAC                          DL0:SU05.TSX                      |
|   DL0:MACRO.SAV                           DL0:ACCESS.TSX                    |
|   DL1:TSKMON.BAK                          DL0:LOGON.SAV                     |
|   RK0:[SYSMGR]SYSMON.NEW                  DL0:SU05B.TSX                     |
|   DL0:DEF.COM                             DL1:TSKMN2.BAK                    |
|   DL0:SYSMON.SAV                          DL0:DUP.SAV                       |
|                                                                             |
+-----------------------------------------------------------------------------+
```

The directory cache display shows what file entries are in the directory cache. This information can be used to determine what files are being used frequently and to determine the best size for the directory cache. Logical disks are shown in square brackets; if the file in the entry is nested more than two logical disks deep, each omitted intervening logical disk is denoted with an extra period.

7.10 Shared file data cache display

```
+-----------------------------------------------------------------------+
|                                                                       |
|                    TSX-Plus SYSMON Utility                            |
|                    25-Jul-84       14:14:06                           |
|                                                                       |
|          *****  Shared File Data Cache Display  *****                 |
|                                                                       |
|                                          Cur        Total             |
|                                                                       |
|          Reads from shared files         240         6818             |
|                                                                       |
|          Reads from data cache           196         4924             |
|                                                                       |
|          Percent reads from cache         81 %        72 %            |
|                                                                       |
|          Writes to shared files           30         3328             |
|                                                                       |
|          Writes through cache             28         3216             |
|                                                                       |
|          Free shared file channels                    25              |
|                                                                       |
|          Blocks in cache (NUMDC)                       0              |
|                                                                       |
+-----------------------------------------------------------------------+
```

The shared file data cache display shows information on utilization of the TSX-Plus file locking facility. The NUMDC parameter controls the number of buffers used for shared file data caching. The I/O counters can be reset using the TSX-Plus monitor RESET command. This may be necessary, as the I/O counts are stored in TSX-Plus as sixteen bit (single word) integers; these tend to overflow after a large amount of processing. The information presented here can be useful in tuning the data cache. This is done by setting NUMDC to various values and observing the speed of the I/O and the percentage of I/O being done out of the data cache. Also, other effects can be noted, such as the effect of two programs doing shared file I/O, and the speed of I/O on various devices.

7.11 Data cache display

```
+-----------------------------------------------------------------------+
|                                                                       |
|                    TSX-Plus SYSMON Utility                            |
|                    25-Jul-84      14:14:06                            |
|                                                                       |
|              ***** Data Cache Display *****                           |
|                                                                       |
|                                                                       |
|                                          Cur        Total             |
|                                                                       |
|        Reads from mounted devices         57         7968            |
|                                                                       |
|        Blocks read from mounted devices  292        55629            |
|                                                                       |
|        Blocks read from cache            292        48468            |
|                                                                       |
|        Percent blocks read from cache    100 %        87 %           |
|                                                                       |
|        Writes to mounted devices         101         4556            |
|                                                                       |
|        Blocks written to mounted devices 103         8051            |
|                                                                       |
|        Blocks updated in cache           103        15212            |
|                                                                       |
|        Data cache size                               1000            |
|                                                                       |
+-----------------------------------------------------------------------+
```

The data cache display shows information on utilization of the TSX-Plus generalized data caching facility. The information presented here can be useful in tuning the data cache. This is done by using the SET CACHE command to enable various cache sizes, and observing the effect on the cache hit ratio. The intent is to allow as much memory for the data cache as may be effectively used, but to also leave as much free memory as possible for user jobs to minimize job swapping. In this context, swapping activity includes both swapping of jobs in and out of memory, and the moving of jobs around in memory to enable all jobs to get the memory they need when job sizes are changing. Note that both the SET CACHE command and the RESET command will reset the cache counters.

7.12 CL device display

```
+-------------------------------------------------------------------------+
|                                                                         |
|                        TSX-Plus SYSMON Utility                          |
|                        27-Jul-84        18:03:41                        |
|                                                                         |
|                    *****  CL Device Display *****                       |
|                                                                         |
| CL Device Number = 2          Assigned to line =  6       Spooled       |
|                                                                         |
| CL SET Options = CR DTR NOFORM NOFORMO LC TAB CRTL LFIN LFOUT NOBININ   |
|                  NOBINOUT                                               |
|                                                                         |
| Line Speed . . . . . . . . 9600      XOFF sent . . . . . . . . .   no   |
|                                                                         |
| Page Length . . . . . . . .   66     XOFF received . . . . . . .   no   |
|                                                                         |
| Line Width . . . . . . . .  132      Read pending . . . . . . .   yes   |
|                                                                         |
| Skip . . . . . . . . . . .    5      Write pending . . . . . .    no   |
|                                                                         |
| Input ring buffer used  .     0      Break being sent . . . .     no   |
|                                                                         |
| In use by job . . . . . .     3   CRAIG                                 |
|                                                                         |
|                                                                         |
+-------------------------------------------------------------------------+
```

The CL device display shows the CL device settings, which job has the device allocated or actually in use, and various device status values. To obtain a cycling display of all the CL devices, type return when prompted for the device number.

7.13 Exiting SYSMON

To exit SYSMON, type RETURN to leave the current display, and type RETURN to the display number prompt. This will cause SYSMON to exit, and will clear the screen.

Appendix A - Startup Error Messages

The following error messages can be displayed during the startup of TSX-Plus. All are fatal error messages and once reported, abort running TSX-Plus. All messages are in the format

?TSX-F-error message displayed here

Cannot find device handler file: dd

The device handler file, "SY:dd.TSX" (where dd represents a two character device driver name), could not be found. A device handler file must exist for each device listed in TSGEN using the DEVDEF declaration. Check the devices specified by the DEVDEF declarations in TSGEN, make any necessary corrections, and generate a new system. Verify that the handler file is present on the system disk and if it is not, place a copy on the system device. Standard device drivers supported by TSX-Plus are supplied on the distribution media. If this is a standard device driver, the file may be copied from the distribution media to the system device. If the device handler is not provided on the distribution media, a device handler file must be built to run with TSX-Plus. (See the System Manager's Guide for information concerning the building of device handlers for TSX-Plus.)

Cannot find "SY:CCL.SAV" file

The file "SY:CCL.SAV" cannot be found. This file was provided on the distribution media and should be copied to the system device.

Cannot locate "SY:SYSODT.REL" file

The file "SY:SYSODT.REL" cannot be located when attempting to run with the system debugger. This file was provided on the distribution media and should be copied to the system device when attempting to install TSX-Plus with the system debugger.

Cannot find "SY:TSKMON.SAV" file
Cannot locate "SY:TSX.SAV"

The file "SY:TSKMON.SAV" or "SY:TSX.SAV" cannot be found. These files are created during the system generation process. If TSX-Plus was not generated on the system device, copy the missing file from the device used to build TSX-Plus to the system device. If the file does not exist on the generation device, the system generation was not successful. (See the TSX-Plus Installation Guide for information concerning system generations.) These files are provided on the distribution media for PRO/TSX-Plus, which does not require system generation; copy these files to the system disk.

Cannot open PLAS region swap file
Number of contiguous blocks needed = nnnnnn

The PLAS (Program Logical Address Space) swap file defined in TSGEN cannot be created because the specified device does not have enough contiguous unused blocks. The number of contiguous blocks required is represented by the number "nnnnnn". Remove any unnecessary files from the disk assigned to contain the PLAS swap file and consolidate the unused blocks (see the SQUEEZE command in the RT-11 User's Guide) until enough contiguous free space is available.

Cannot open program swap file
Number of contiguous blocks needed = nnnnnn

The program swap file defined in TSGEN cannot be created because the specified device does not have enough contiguous unused blocks. The number of contiguous blocks required is represented by the number "nnnnnn". Remove any unnecessary files from the disk assigned to contain the swap file and consolidate the unused blocks (see the SQUEEZE command in the RT-11 User's Guide) until enough contiguous free space is available.

Cannot open shared run-time file: dev:file.ext

The shared run-time file named "dev:file.ext" specified in TSGEN could not be opened. Check the run-time file names defined in TSGEN, make any necessary corrections, and generate a new system. Verify that the file is present on the device specified and if it is not, place a copy on that device.

Cannot open spooled device: dd

The spooled device named "dd" cannot be opened by TSX-Plus. Check the device names specified in the SPOOL declaration in TSGEN, make any necessary corrections, and generate a new system. Verify that the device handler file "SY:dd.SYS" is present on the system device and if it is not, place a copy on that device and install the device handler in RT-11 (see the INSTALL command in the RT-11 User's Guide).

Cannot open SY:INDTMP.TSX file. # blocks needed = nnnnnn

The IND storage file named "SY:INDTMP.TSX" cannot be created because the system device does not have enough contiguous unused blocks. The number of contiguous blocks required is specified by the number "nnnnnn". Remove any unnecessary files from the system disk and consolidate the unused blocks (see the SQUEEZE command in the RT-11 User's Guide) until enough contiguous free space is available.

Cannot open TSXUCL data file. # blocks needed = nnnnnn

The UCL (User Command Language) data file named "SY:TSXUCL.TSX" cannot be created because the system disk does not have enough contiguous unused blocks. The number of contiguous blocks required is specified by the number "nnnnnn". Remove any unnecessary files from the system disk and consolidate the unused blocks (see the SQUEEZE command in the RT-11 User's Guide) until enough contiguous free space is available.

Error on read of SYSODT rel file

An input error occurred when reading the system debugger file named "SY:SYSODT.REL" into memory. Check the system disk and the hardware involved.

Error reading device handler file: dd

An input error occurred when reading the device handler file "SY:dd.TSX" into memory. Check the system disk and the hardware involved.

Error reading "SY:TSX.SAV"

An input error occurred when reading the memory resident overlays from the file "SY:TSX.SAV" into memory. Check the system disk and the hardware involved.

Generated TSX system is too large

The TSX-Plus system is generated too large to load and run. Although this may occur when there is not enough total physical memory, it usually implies that the unmapped portion of TSX-Plus exceeds 40Kb. Remove any unnecessary features, decrease excessive parameters, and generate a smaller system. See the TSX-Plus Installation Guide for information on the effect of optional features on the size of TSX-Plus.

Handler for SY device was not loaded

The handler for the system device was not defined using a DEVDEF declaration in TSGEN. Specify the system device handler with a DEVDEF declaration in TSGEN and generate a new system.

Handler not generated with extended memory support: dd

The device handler file named "SY:dd.TSX" (where "dd" represents the two character device driver name) was not generated with the required memory management support. Verify that the device handler was written to include support for memory management and review the TSX-Plus System Manager's Guide for building device drivers for TSX-Plus.

Insufficient disk space for spool file

The spool file defined in TSGEN cannot be created because the specified disk does not have enough contiguous unused blocks. The number of required contiguous blocks was defined using the SPOOL declaration in TSGEN. Either decrease the number of blocks required; or remove any unnecessary files from the disk assigned to contain the spool file and consolidate the unused blocks (see the SQUEEZE command in the RT-11 User's Guide) until enough contiguous free space is available.

Insufficient memory space for data cache

Not enough memory was available to load TSX-Plus and allocate the specified number of data cache blocks defined by the CACHE parameter in TSGEN. Decrease the number of data cache blocks specified or install more memory.

Insufficient memory to load all mapped system regions

Not enough memory was available to load memory resident mapped system regions in TSX-Plus. Review the Software Product Description to determine if you have sufficient memory. Review the system generation parameters in TSGEN, remove any unnecessary features, decrease excessive parameters and generate a smaller system or install more memory.

Insufficient memory to load all shared run-time systems

There is not enough memory to load all the shared run-time systems specified in TSGEN. Review the system generation parameters in TSGEN, remove any unnecessary features, decrease excessive parameters (remove some or all of the shared run-time systems) and generate a smaller system or install more memory.

Insufficient total physical memory for generated system

The TSX-Plus system generated with the specified features was too large for the physical memory available. Review the system generation parameters in TSGEN, remove any unnecessary features and decrease excessive parameters, and generate a smaller system or install more memory.

Invalid interrupt vector address for T/S line:
Line # = nn

The line numbered "nn" has an invalid vector address. Time-sharing lines may only be vectored in the range 60 to 477, and may not use a vector assigned to any other device. Lines are numbered sequentially in increasing order by LINDEF specification in TSGEN. Correct the vector address for the indicated line and generate a new system.

Invalid RT-11 version for device handler dd

A device handler named "dd" (where "dd" represents a two character device driver name) required a later version of RT-11 than the one installed on the system device. During initialization, TSX-Plus analyzes each device defined by the DEVDEF specification in TSGEN and determines if the present RT-11 version will support the inclusion of that handler. Update RT-11 to the correct version which supports the device specified.

Invalid status register address for T/S line:
Line # = nn

The line numbered "nn" has an invalid status register. When the specified CSR is addressed by the CPU, the location does not respond. Lines are numbered sequentially in increasing order by LINDEF specification in TSGEN. Correct the status register for the indicated line and generate a new system.

RT-11 doesn´t recognize device: dd

The device handler named "dd" (where "dd" represents a two character device driver name) is not recognized by RT-11 and therefore cannot be loaded by TSX-Plus. Install the device handler in RT-11 (see the INSTALL command in the RT-11 User´s Guide).

System is not equipped with extended memory management hardware

TSX-Plus could not find extended memory management hardware. Set the TSGEN parameter EXTMCH to zero (0) and generate a new system or install extended memory management hardware.

System is not equipped with memory management hardware

TSX-Plus will not run on the current hardware configuration because it requires memory management support. Check the TSX-Plus Software Product Description to determine if any other required hardware is necessary and install the necessary hardware.

TSX generation did not include device dd

One of the files defined in TSGEN included a device handler named "dd" (where "dd" represents the two character device driver name) that was not specified in the handler declaration section DEVDEF in TSGEN. Check all file specification and DEVDEF declarations in TSGEN, make any necessary corrections, and generate a new system.

TSX is already running

TSX-Plus is currently running and therefore cannot be started again.

Appendix B - SYSTEM ERROR MESSAGES

The error messages detailed below are reported when fatal system errors occur and once displayed on the system console the operating system will halt. All of the system error messages have identical formats which display the following information:

```
<BELL>?TSX-F-Fatal system error at nnnnn
EEE-Error message displayed here (see below)
Arg. value = nnnnnn
```

Depending on "Arg. value = nnnnnn", additional information may be displayed. If the argument value is below 120000, no additional text is displayed; only the above three lines appear. When the argument value is 120000 or higher, additional information will be provided indicating the identity of the system region mapped when the error occurred. This additional information will take one of three formats.

The following additional information is displayed when the mapped system region points to a mapped device handler:

```
Device name: xxx
```

The following additional information is displayed when the mapped system region points to a memory resident code overlay region:

```
Seg. value = nnnnnn
Overlay: xxx
```

The following additional information is displayed when the mapped system region does not point to either a mapped device handler or a memory resident code overlay region:

```
PAR5 value = nnnnnn
```

DTL-Demonstration system time limit reached

The time limit has expired for the demonstration system. This time limit is generally thirty minutes. TSX-Plus may be started again.

FRK-No free FORK blocks

A system routine issued a FORK request when the FORK queue was full. One or more devices is repeatedly interrupting faster than can be processed by the system.

INO-Interrupt occurred at location 0

The hardware has asserted an interrupt at location 0 where no device resides.

KRE-KMON read error

An input error occurred when attempting to read the file "SY:TSKMON.SAV" into memory. This indicates a hardware malfunction was reported by the handler.

KTP-Kernel mode trap

A trap through vector 4, 10, or 250 occurred in kernel mode while at interrupt level. The argument value indicates the address at which the trap occurred. If the trap address was 120000 or higher, the additional information specifies the overlay code section which was mapped when the error occurred.

LMF-Job lock mem failure

A system failure occurred when no memory was available in which to lock a job that had previously requested memory.

MIO-Need to increase value of MIONWB sysgen parameter

The system attempted to perform an I/O operation to a device that requires I/O mapping and there were no free system I/O mapping buffers or wait queue elements. You must increase either the MIONBF parameter which will allocate more I/O mapping buffers or the MIONWB parameter that increases the number of wait queue elements.

MPR-Memory parity error

A trap occurred through vector 114 indicating a hardware memory parity error was detected.

NQE-Ran out of free I/O queue elements

An attempt was made to queue a system I/O request and no I/O queue elements were available.

PFT-Power-fail trap

A trap occurred through vector 24 indicating a hardware power failure.

RIT-Trap in real-time interrupt service routine

A trap in a real-time interrupt service routine causes a system halt because interrupt service routines run at fork level. A trap in a real-time interrupt completion routine does not cause a system halt.

SIE-Swap file I/O error

An input or output error occurred either reading or writing into the program swap file.

SJN-Job # 0 at STOP

A job number of zero was detected during a request to stop the current job and execute "SY:TSKMON.SAV". User job numbers must be greater than zero.

SOF-Stack overflow

One of the three system stacks has overflowed. The argument value indicates which stack has overflowed.

SSE-PLAS region swap file I/O error

A hardware I/O error was detected while reading or writing a PLAS region to the swap file. The device used for the PLAS swap file is specified in TSGEN with the RSFBLK parameter.

UEI-Interrupt occurred at unexpected location

An interrupt occurred through a vector that was not attached to a terminal or CL line, a device handler, or a real-time completion routine. The argument value indicates the vector location.

Index